UNIVERSITÀ DEGLI STUDI DI PISA

Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea in Scienze dell'Informazione

Tesi di Laurea

Progettazione e realizzazione di un servizio interattivo per la Pubblica Amministrazione

Candidati:
Giovanni Fleres e Alessandro Rontani

Relatori:

Prof. Vincenzo Ambriola

Ing. Paolo Fiorenzani

Dott. Simone Pierazzini

Controrelatore:

Prof. Antonio Albano

Anno Accademico 2002/2003

Sommario

Indice delle figure

Indice delle tabelle

Ai nostri genitori,
ad Annamaria,
a Davide e a Simone.
Per la loro pazienza.

A Paolo, Giacomo, Ioletta, Alessandra,
Federica, Simone, Roberto, Luca, Sergio
e a tutte le altre persone di Metaware che
ci hanno aiutato e sostenuto durante questo
lavoro.

Capitolo 1: Introduzione

Area di riferimento

La Pubblica Amministrazione, per sua natura, si è sempre trovata a dover gestire un quantitativo molto grande di informazioni attraverso differenti procedure. L'enormità di questa base di conoscenza è stata la principale causa alla base dell'inefficienza che ha caratterizzato molte branche della Pubblica Amministrazione in passato, ma in tempi recenti l'avvento dell'informatizzazione a basso costo ha reso economicamente possibile l'automatizzazione di queste procedure. Il processo di trasformazione delle relazioni interne ed esterne della pubblica amministrazione che attraverso l'utilizzo di tecnologia informatica e di comunicazione punta ad ottimizzare l'erogazione dei servizi, a incrementare la partecipazione di cittadini e imprese e a migliorare la capacità di governo della stessa pubblica amministrazione, è stato definito come “e-government”[Savona03] e i suoi campi di applicazione si sono estesi in un vasto spettro di utilizzi. Alcuni esempi possono essere la digitalizzazione di documenti cartacei per la consultazione elettronica, database che contengono vari dati relativi ai cittadini e l'utilizzo delle nuove tecnologie per il voto elettronico.

Area specifica

La diffusione dell'informatica a basso costo ed il numero sempre crescente di utenti connessi alla rete Internet hanno reso possibile la nascita di un vasto numero di servizi interattivi rivolti all'utente, soprattutto da parte di aziende private, come banche, negozi o produttori. Alla luce degli ottimi risultati, sia in termini di efficienza che in termini di soddisfazione dell'utente, anche la pubblica amministrazione ha iniziato a fornire alcuni dei suoi servizi attraverso sistemi informatici interattivi e l'intento dei legislatori è che quanto prima ogni servizio della pubblica amministrazione possa

essere fruito in modo interattivo. I vantaggi di un servizio interattivo sono molteplici:

- Maggiore efficienza: fornendo la possibilità agli utenti di usufruire del servizio attraverso una postazione Internet vengono eliminati sia i tempi morti costituiti dalle file allo sportello, sia quelli che il cittadino disperde per recarsi fisicamente allo sportello.
- Maggiore trasparenza: l'utente può avere l'immediata percezione di quale sia lo stato dei procedimenti avviati.
- Maggiore semplicità: attraverso la disponibilità di strumenti che aiutano nella compilazione, un servizio interattivo può semplificare in maniera significativa la compilazione esatta di moduli e richieste da parte del cittadino.

Problema affrontato

Nell'ambito dei servizi interattivi per la pubblica amministrazione, il problema specifico su cui abbiamo deciso di focalizzare la nostra attenzione è quello dello Sportello Unico per le Attività Produttive (SUAP). L'idea di SUAP è stata introdotta nella legislazione italiana con il DPR n. 447 del 20 Ottobre 1998, ed è definito come “un interlocutore unico per le imprese per tutto ciò che attiene alla localizzazione, alla costituzione ed alla ristrutturazione di impianti produttivi”. La virtualizzazione di uno sportello unico comporta la risoluzione di una vasta gamma di problemi, che non sono completamente risolvibili in modo efficiente con gli strumenti disponibili, che non riescono ad amalgamare in maniera semplice e indipendente dalla piattaforma le tecnologie necessarie a gestire correttamente uno sportello unico interattivo, come knowledge management, content management, web publishing, gestione dei diritti e gestione della collaborazione.

Soluzioni esistenti e loro limiti

La strada seguita nella maggior parte delle soluzioni attualmente esistenti, è stata quella di adattare le applicazioni esistenti alla

necessità di offrire servizi interattivi al cittadino e alle imprese. Questa scelta ha sicuramente comportato un risparmio in termini di risorse economiche e tecnologiche, ma il fatto che queste applicazioni non fossero state progettate esplicitamente per queste esigenze, ha limitato fortemente l'usabilità dei servizi. Il maggiore di questi limiti è dato sicuramente dalla scarsa integrazione dei moduli di back office e front office, derivante dal fatto che originariamente le applicazioni che si occupano di questi due ambiti sono state spesso progettate in maniera indipendente l'una dall'altra.

La soluzione proposta e i suoi vantaggi

La nostra soluzione supera i limiti delle applicazioni esistenti fornendo un framework, indipendente dal sistema operativo e basato sul web, che permette agli strumenti di back office di integrarsi strettamente con quelli di front office in maniera trasparente. Il framework, infatti, integra funzionalità di content e knowledge management, di web publishing, di gestione della collaborazione, ed è stato sviluppato da zero, senza basarsi su applicativi esistenti. Inoltre sono stati utilizzati, dove era possibile, protocolli e formati standard quali XML, HTTP, PDF e XML:DB.

Documentazione di progetto

I requisiti fondamentali del sistema sviluppato sono: indipendenza dal sistema operativo, accessibilità attraverso un qualsiasi browser web, interoperabilità tra le diverse applicazioni sviluppate con il framework, supporto alla modularità delle applicazioni, gestione a granularità variabile dei diritti, basso costo. A causa di quest'ultimo requisito sono state privilegiate soluzioni Open Source, o comunque a basso costo, che aderissero quanto più possibile ad uno standard. Java è stato il linguaggio di programmazione adottato, a causa soprattutto della sua indipendenza dal sistema operativo e della sua stabilità. Per i documenti si è scelto di utilizzare il formato XML, che consente di integrare informazioni semantiche assieme al contenuto,

e questo ha guidato la scelta di utilizzare come repository della base di conoscenza database XML nativi che seguono lo standard XML:DB.

Struttura della tesi

Nel capitolo 2 verrà presentato il dominio trattato, attraverso una panoramica del problema che presenta la legge sullo sportello unico e alcuni scenari e casi d'uso. Nel capitolo 3 verrà presentato quello che è lo stato dell'arte dei servizi interattivi per la Pubblica Amministrazione, con una particolare attenzione ai servizi SUAP. Nel capitolo 4 verranno presentate le tecnologie utilizzate e le motivazioni che hanno portato a scegliere determinate soluzioni piuttosto che altre. Nel capitolo 5 verrà illustrata la soluzione proposta attraverso una descrizione delle funzionalità offerte dal framework e delle scelte di progetto fatte durante lo sviluppo. Nel capitolo 6 sarà presentato il modulo di “comunicazione di avvio”, un esempio concreto di applicazione costruita attraverso il framework che integra molte delle funzionalità fornite da quest'ultimo. Nel capitolo 7 saranno presentate le conclusioni e i problemi rimasti aperti.

Capitolo 2: Una panoramica del problema

In questo capitolo si illustreranno i vantaggi legati all'introduzione dell'informatica nella pubblica amministrazione e le problematiche che si incontrano in questo ambito. Sarà data una particolare enfasi ai problemi riscontrati nell'ambito della gestione di uno Sportello Unico per le Attività Produttive e saranno infine illustrati alcuni possibili scenari in cui un sistema informatico aumenta l'efficienza delle procedure burocratiche

Cos'è l'e-government

La parola e-government può assumere una grande varietà di significati, a seconda del livello di informatizzazione implicato, spaziando dalla semplice acquisizione di un documento cartaceo in forma digitale sino ad un complesso sistema informativo che automatizzi un'intera procedura amministrativa. L'Unione Europea ha definito quattro livelli di interazione per quanto riguarda i servizi forniti dalla pubblica amministrazione [ConsMin02]:

1. Livello informativo: a questo livello sono disponibili solo le informazioni necessarie e sufficienti per avviare la procedura che porta all'erogazione del servizio. I vantaggi sono che il livello informativo è ottenibile con un minimo sforzo e che gli utenti possono reperire da soli, senza il bisogno di alcun operatore, le informazioni necessarie attraverso un unico punto di accesso. Lo svantaggio principale di questa implementazione è che il guadagno di efficienza per lo svolgimento della pratica è minimo, visto che è equivalente a consultare la documentazione cartacea allo sportello.
2. Download modulistica: in questo caso è possibile scaricare online i moduli necessari ad avviare la procedura che porta all'erogazione del servizio (interazione one-way). Il vantaggio principale di questa implementazione è che l'utente non deve recarsi fisicamente presso l'ufficio per ritirare i moduli, ma può compilarli con comodo a casa (dopo averli stampati o ordinati via posta) e rispedirli

all'ufficio competente, anche per posta. Ovviamente sorge il problema del formato in cui sono distribuiti questi moduli (bisogna assicurarsi che l'utente sia in grado di visualizzarli e stamparli) ed inoltre rimane il fatto che l'utente è sempre legato alla modulistica cartacea e ai tempi di smistamento della posta ordinaria, senza considerare i costi aggiuntivi dati da carta, inchiostro e servizio postale.

3. Inoltro richiesta: A questo livello si ha un'interazione two-way, si può, cioè, avviare direttamente online la procedura che porta all'erogazione del servizio, attraverso modulistica elettronica e firma digitale. Il vantaggio più evidente a questo livello è il risparmio di tempo dell'utente, che non deve attendere che il modulo arrivi all'ufficio, ma avvia immediatamente la procedura. Inoltre il sistema può prevedere degli algoritmi di verifica sulla correttezza dei dati inseriti, in modo da evitare rallentamenti nella pratica dovuti ad errori di compilazione. Tra gli svantaggi principali rimane il fatto di essere ancora un sistema ibrido, che può implicare comunque l'uso di moduli cartacei che rallentano la procedura, visto che l'unica fase gestita digitalmente è quella del front office.
4. Esecuzione transazione, compreso pagamento e consegna: A quest'ultimo livello di interazione tra cittadino e pubblica amministrazione ogni fase della procedura viene eseguita da un sistema informatico, che elimina completamente il bisogno di moduli cartacei e minimizza la necessità di intervento da parte di un operatore. In questo caso si ha il massimo livello di efficienza nello svolgimento della pratica, dato che ogni operazione del sistema avviene in un tempo praticamente istantaneo e gli unici tempi morti sono quelli dovuti alle scadenze e ai periodi temporali previsti dalle leggi vigenti. Inoltre il risparmio di risorse umane e materiali (carta, inchiostri etc.) è ancora più elevato rispetto al livello precedente C'è da dire comunque che un sistema di questo

genere ha, ovviamente, un certo costo iniziale (in termini di formazione e addestramento del personale) non indifferente, ma si ritiene che questo costo possa venire ammortizzato nel medio periodo.

Lo sportello unico

Introduzione

Nell'ambito dell'e-government e, in particolare, nel settore dei servizi interattivi, la pubblica amministrazione italiana ha deciso di investire una notevole quantità di risorse nella realizzazione dello Sportello Unico per le Attività Produttive (SUAP): il tema della semplificazione e dello snellimento delle procedure amministrative rappresenta oggi una grande occasione per accrescere la competitività del settore produttivo e per incrementare lo sviluppo e il rinnovamento dell'intero sistema economico del paese. Nel corso degli ultimi anni, grazie soprattutto alle "leggi Bassanini" e all'azione del Nucleo per la semplificazione delle Norme e Procedure, si è intrapreso un complesso programma di riforme, tendenti a restituire all'apparato amministrativo la smarrita credibilità da parte dei cittadini e del mondo produttivo. Tale obiettivo è stato raggiunto attraverso la devoluzione di funzioni e compiti amministrativi dallo Stato agli enti locali, ritenuti più solleciti nei confronti delle istanze dei privati proprio per la loro diretta presenza sul territorio.

La legislazione vigente

Il conferimento di funzioni e competenze amministrative dallo Stato agli enti locali è stato disposto con la legge delega 15 Marzo 1997, n. 59, in attuazione della quale è stato successivamente emanato il D.Lgs 31 marzo 1998, n. 112 per la parte concernente le attività produttive ed il D.Lgs 31 Marzo 1998 n.143 per quella relativa all'internazionalizzazione delle imprese. È da precisare che, in entrambi i casi, la legge delega subordinava il riordino delle funzioni all'individuazione "per quanto possibile..[di] momenti decisionali

unitari.."(art,4, lett. c) avviando per questo verso anche una contestuale semplificazione dell'organizzazione e delle procedure amministrative. In particolare è nel capo IV° del D.Lgs m.112/98 che viene introdotto per il prima volta l'idea di "Sportello Unico per le Attività Produttive", che è stata poi disciplinata dal DPR 20 Ottobre 1998, n. 447, come modificato dalla legge 24 novembre 2000, n.340 nonché, più estesamente, dal DPR 7 Dicembre 2000, n.400. Lo scopo principale di questo regolamento è quello di costituire finalmente un interlocutore unico per le imprese per tutto ciò che attiene alla localizzazione, alla costituzione ed alla ristrutturazione di impianti produttivi. Franco Bassanini, allora Ministro per la Funzione Pubblica, spiegando cosa sia e perché abbia progettato lo Sportello Unico per le Attività Produttive ha detto: “Lo Sportello Unico per le Attività Produttive è un'innovazione introdotta nell'ambito della riforma delle pubbliche amministrazioni che consente a chi vuole avviare una nuova attività produttiva, per esempio costruire uno stabilimento o costruire un laboratorio artigiano, di avere un unico interlocutore nei sempre più numerosi comuni italiani che si stanno attrezzando e applicheranno questa riforma. Oggi chi vuole costruire uno stabilimento deve ottenere 20, 30 e in, qualche caso, sino a 43 diverse autorizzazioni, nulla osta, licenze, concessioni; insomma atti dell'amministrazione. Da marzo, nei comuni che hanno attivato lo sportello unico, l'imprenditore si rivolge allo sportello, fa un'unica domanda, presenta un unico progetto e in un tempo molto più rapido che non in passato ha un'unica risposta, che può essere naturalmente positiva o negativa in base alle vigenti disposizioni per aprire uno stabilimento, un laboratorio artigiano, un albergo, un esercizio commerciale. Se la risposta è positiva, però, è una concessione edilizia che contiene tutte le autorizzazioni necessarie. Il procedimento prevede nei casi più semplici una risposta che arriva in tre mesi, mentre nei casi più complicati, dove è necessaria una valutazione di impatto ambientale, in 11 mesi. A differenza dei tempi

attuali dove spesso occorrono anni."[1] Va detto che la semplificazione normativa, introdotta dallo sportello unico nel nostro ordinamento, è stata concepita, oltre che per il motivo che abbiamo detto, anche per produrre una significativa riduzione dei costi amministrativi che gravano sulle imprese italiane, soprattutto piccole e medie, costi che peraltro costituiscono fonte di discriminazione delle stesse nei confronti delle imprese dei altri paesi europei. Consentire al sistema produttivo nazionale di affrontare la concorrenza del mercato unico europeo, rappresenta quindi la ragione principale che sostiene tutta la normativa sullo Sportello Unico per le Attività Produttive.

Quali cambiamenti produce il SUAP nel rapporto tra la pubblica amministrazione e le imprese

Prima dell'entrata in vigore delle leggi che introducono il regolamento concernente il SUAP, un imprenditore che intendeva aprire una qualsiasi attività doveva presentare le relative istanze ai vari uffici competenti, essendo quindi a suo carico l'onere di portare la documentazione da un ente all'altro. La riforma dello sportello unico attribuisce invece ai comuni le funzioni amministrative concernenti la realizzazione, l'ampliamento, la cessazione, la riattivazione, la localizzazione e la rilocalizzazione di impianti produttivi, ivi incluso il rilascio delle concessioni o autorizzazioni edilizie. L'imprenditore avrà pertanto la possibilità di presentare un unico procedimento amministrativo in materia di autorizzazione all'insediamento di attività produttive. Inoltre viene creata la figura del responsabile dello sportello unico che sarà il responsabile unico dell'intero procedimento. Il Procedimento Unico, che l'imprenditore attiva presso il Comune, è quindi composto dalle singole procedure distinte che coinvolgono le materie e gli adempimenti di competenza di enti ed uffici diversi che devono essere espletati per l'attivazione di una determinata attività produttiva.

1 http://www.repubblica.it/online/speciale/media/media/media.html

I soggetti coinvolti nel SUAP

Quali sono le entità che interagiscono con il SUAP

Ai servizi dello sportello unico accedono tre categorie di utenti: l'utente finale (ovvero l'imprenditore), che è colui che è interessato ad avviare domande per procedimenti autorizzatori, a conoscere i relativi meccanismi e ad essere informato sullo stato di avanzamento della pratica; gli enti, interessati a scambiarsi documenti relativi ad uno stesso procedimento e ad essere aggiornati sullo stato di avanzamento delle pratiche di competenza; il responsabile dello sportello unico, che deve disporre di uno strumento per controllare la gestione del procedimento unico e coordinare lo scambio di documenti e atti fra tutte le parti coinvolte.

Utente finale

Per utente finale si intende colui che per avviare una pratica relativa ad un'attività produttiva, entra in contatto con il responsabile dello sportello unico del comune di competenza. In questa prima fase il responsabile individua la documentazione completa che l'utente deve allegare alla domanda d'avvio del procedimento (per esempio le varie autocertificazioni). Una volta ricevuta l'intera documentazione, il responsabile può avviare il procedimento, e le informazioni relative al suo stato di avanzamento devono essere accessibili all'utente.

Responsabile dello Sportello

Il Responsabile, oltre ad aiutare l'utente nelle fasi istruttorie della pratica, definendo il flusso informativo del procedimento che deve essere avviato, individuando gli enti coinvolti nella valutazione della domanda, i relativi adempimenti, i dettagli documentali e temporali dei relativi passaggi, si occupa anche di:

- Controllare le scadenze previste dal flusso.
- Rendere accessibile al cittadino e a quanti altri competano le informazioni sullo stato di avanzamento del procedimento in corso.
- Rendere pubbliche, nei tempi e nei modi previsti dalla legge, le

informazioni sul procedimento ai “portatori di interessi diffusi”.

Responsabile di enti terzi

Gli enti, tramite lo Sportello Unico, accedono alle necessarie informazioni relative alla pratica e restituiscono, nei tempi e nei modi previsti dalla normativa, la documentazione richiesta per la prosecuzione del procedimento.

Requisiti del SUAP

Lo Sportello Unico, per adempiere ai propri compiti istituzionali[2], deve quindi offrire gli strumenti per gestire:

- una base di conoscenza di attività produttive, procedure, normative, modulistica;
- i rapporti con il cittadino-imprenditore;
- i rapporti intersoggettivi: gli scambi di documentazione, pareri ed atti con gli enti terzi coinvolti nel procedimento unico;
- i rapporti interorganici con gli altri uffici della struttura comunale;
- l'iter del procedimento unico:
  - preistruttoria
  - presentazione della domanda unica
  - gestione del procedimento
  - rilascio dell'atto finale
  - adempimenti successivi
  - eventuali verifiche.

Per realizzare lo Sportello Unico servono due tipologie di software: quella per il back office e quella per il front office. La prima dovrà permettere sia la gestione dell'intera istruttoria delle pratiche attivate con le Domande Uniche che la gestione dei rapporti tra gli enti pubblici coinvolti nel Procedimento Unico attraverso lo scambio

2 “Lo sportello unico, previa predisposizione di un archivio informatico contenente i necessari elementi informativi, a chiunque vi abbia interesse, l'accesso gratuito, anche in via telematica, alle informazioni sugli adempimenti necessari per le procedure previste dal presente regolamento, all'elenco delle domande di autorizzazione presentate, allo stato del loro iter procedurale, nonché a tutte le informazioni utili disponibili a livello regionale comprese quelle concernenti le attività promozionali”[DPP447]

di documenti, richiesta di pareri o atti, rilascio risposte secondo le peculiari competenze. Le applicazioni del front office dovranno invece permettere all'imprenditore di ricevere le comunicazioni da parte dell'ufficio in tempo reale online e di essere aggiornato sull'evoluzione e l'esito dell'istruttoria potendo consultare lo stato delle pratiche da lui attivate.

Scenari e casi d'uso

I procedimenti necessari per un sistema di gestione di uno Sportello Unico per le Attività Produttive costituiscono un'architettura complessa, che implica un vasto spettro di procedure di gestione dei documenti. Per ognuna di queste procedure verrà fornita una descrizione del problema e una valutazione delle difficoltà che si prevedono a seguito di un processo di informatizzazione della procedura. Per ciascuno di questi scenari, comunque, sono richiesti dei requisiti comuni: innanzitutto ognuna delle funzionalità fornite deve essere accessibile da un normale browser web, sia per evitare di costringere l'utente ad installare software specifico per usufruire del servizio (si suppone, ovviamente, che se un utente ha accesso alla rete possieda già un browser), sia per consentire la possibilità di utilizzare il servizio senza essere legati a particolari postazioni d'accesso abilitate specificamente per quell'uso. Un altro requisito fondamentale di ogni sottomodulo del sistema è quello di consentire una verifica automatica dei dati inseriti, per quanto è possibile, in modo da informare l'utilizzatore di eventuali errori nella compilazione. Inoltre, sempre per garantire un'ulteriore sicurezza sull'esattezza dei dati inseriti in documenti che comunque avranno valore legale, il sistema deve fornire all'utilizzatore, al termine di ogni operazione di modifica, una vista con i dati inseriti e la possibilità di confermare l'operazione, modificando effettivamente il documento, o di ritornare alla maschera di inserimento dati per effettuare delle correzioni. Infine, per ragioni di usabilità, un ultimo

requisito è che, nelle varie fasi di una sequenza di editing, i campi delle maschere di inserimento mantengano gli ultimi valori inseriti, in modo da non doverli riscrivere ogni volta.

Creazione domanda unica

La creazione di una “domanda unica” è un esempio di uno scenario abbastanza comune nel campo del Knowledge Management: la creazione guidata e verificata di un documento a partire da frammenti di altri documenti. In questo caso si tratta di riempire un modulo che, a seconda dei valori nei campi compilati, allega diversi documenti per l'autocertificazione e ne produce altri per l'invio agli enti terzi. In questo modulo, oltre ai dati anagrafici necessari a norma di legge, l'utente deve inserire l'attività produttiva oggetto della domanda presentata e il tipo di operazione che intende svolgere (apertura, chiusura, ampliamento, etc.). A seconda dei valori di questi due campi vengono automaticamente aggiunte al documento le descrizioni degli endoprocedimenti, cioè delle procedure da attivare, e vengono inseriti gli allegati relativi agli endoprocedimenti stessi. L'utente deve avere comunque la possibilità di inserire altre procedure (scelte da una base di conoscenza fornita dal sistema) per le sue esigenze specifiche. In questo modo ogni imprenditore ha a disposizione una domanda unica personalizzata per le proprie esigenze.

Comunicazione di avvio

Dopo aver completato la domanda unica, all'utente deve essere rilasciato un documento, chiamato “comunicazione di avvio” che certifica l'avvio del procedimento unico richiesto dall'imprenditore. All'interno di questo documento devono essere presenti le informazioni relative alla domanda unica presentata, i dati anagrafici di chi ha presentato la domanda, i dati relativi all'ufficio presso cui è stata presentata la domanda ed i dati relativi al responsabile incaricato di seguire il procedimento.

Rilascio dell'atto finale

Una volta che tutti i pareri sono stati rilasciati dagli enti competenti, all'imprenditore deve essere rilasciato un atto che certifica l'avvenuta o mancata autorizzazione. Questo atto deve comprendere tutti i responsi degli enti terzi responsabili delle procedure autorizzative, comprendenti le eventuali ragioni che hanno portato ad una mancata autorizzazione.

Agenda appuntamenti, news e annotazioni

Nella pubblica amministrazione non esiste solamente una parte “statica” di conoscenza, come può essere una base documentale legislativa, ma di solito sono presenti anche alcune informazioni che vengono generate “dinamicamente”, nel senso che sono informazioni utili solo per un determinato periodo di tempo. Queste informazioni possono essere annotazioni legate ad una particolare procedura oppure avere un carattere più generale. In quest'ultimo caso, è utile distinguere tra appuntamenti, che trattano di argomenti posizionati in modo preciso nello spazio e nel tempo, e news, che trattano di argomenti in modo più generale. Questa ulteriore suddivisione è necessaria per il diverso tipo di informazione contenuta nel documento: gli appuntamenti devono forzatamente contenere al loro interno le coordinate spazio-temporali dell'evento, mentre per le news queste informazioni non sono necessarie. Tutti i documenti di tipo dinamico sono creati da particolari utenti abilitati, ed è un requisito fondamentale che il documento prodotto contenga il nome dell'autore, in modo da poter rintracciare eventuali responsabilità sul contenuto. Oltre a questo è necessaria anche la figura di uno o più “editori” che possono intervenire sui documenti inseriti per cancellarli o correggerli in caso di inesattezze

Gestione degli utenti e della sicurezza

La pubblica amministrazione è un ambito in cui è presente una fitta suddivisione di ruoli, sia orizzontale che verticale, e questa

complessità si ripercuote sulle operazioni che ogni utente può compiere su determinati documenti: ad esempio, un certo funzionario potrebbe essere autorizzato a modificare le normative riguardanti le concessioni edilizie, ma potrebbe non avere l'autorizzazione a modificare quelle relative ai regolamenti sanitari. È quindi necessaria una gestione dei permessi che permetta di operare con una grana molto fine a livello delle singole azioni che un utente è abilitato a compiere, implementando quella che è diventata una pratica standard nell'ambito della sicurezza: il “principio dei privilegi minimi”[3], secondo cui in un ambiente sicuro ogni elemento del sistema deve avere il minimo insieme di permessi necessari per il compito che deve svolgere. Per facilitare l'assegnazione dei permessi, però, deve anche essere possibile definire dei ruoli generali, che specifichino un insieme di permessi: in questo modo non occorre specificare tutti i permessi concessi ad un singolo utente, ma basta assegnargli uno o più ruoli adatti ai compiti che è abilitato a svolgere. Come ultimo requisito, visto che ogni utente è direttamente responsabile dei documenti che gestisce, si rende necessaria un'anagrafica molto dettagliata dell'utente, in modo che chi ha diritti per visualizzare le informazioni legate ad una certa persona possa rintracciarla in caso di eventuali inesattezze.

Navigazione della base di conoscenza

Uno dei vantaggi fondamentali di un sistema informativo per la gestione documentale è quello di poter organizzare i dati in modo che siano facilmente utilizzabili, ed è immediato osservare che la vasta base di conoscenza legislativa può essere organizzata in modo naturale secondo uno schema ad albero, in cui ogni nodo non terminale rappresenta un insieme di documenti che riguardano lo stesso ambito (“materia”). Questa organizzazione logica si riflette in una modalità di navigazione in cui è visibile una lista di materie da

3 per maggiori dettagli: [FerraioloKuhn92] oppure, per una definizione di "least access privilege", si può consultare [Schneider03]

cui si può accedere alle liste di sottomaterie o di normative al livello sottostante dell'albero.

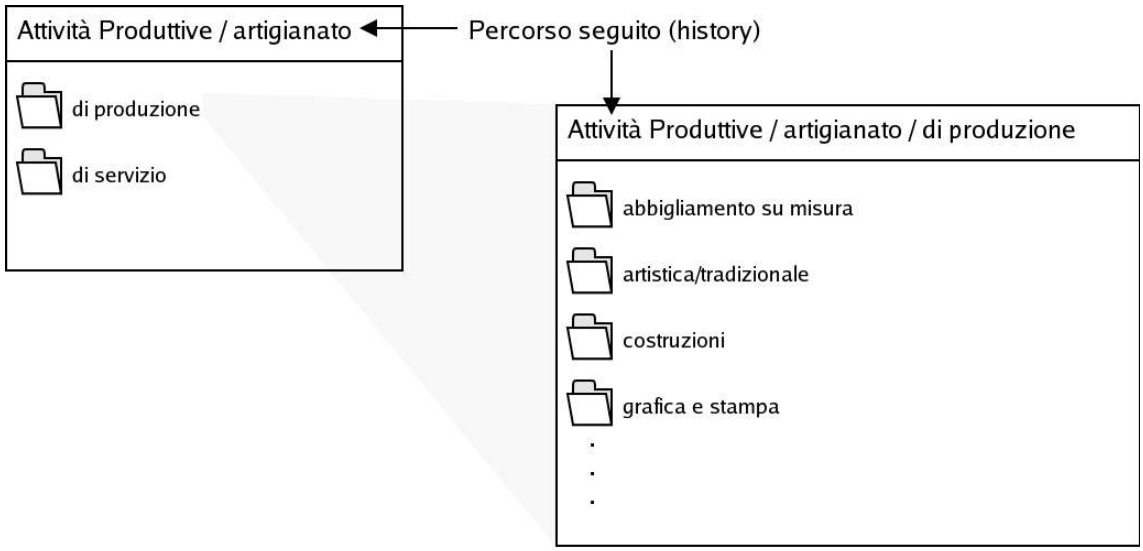

Figura 1: schema della navigazione

Per comodità di navigazione, un ulteriore requisito richiesto è quello di avere una rappresentazione del percorso seguito lungo l'albero, che consenta di tornare ad uno qualsiasi dei livelli successivi con un unica operazione che sfrutti dei collegamenti ipertestuali. Oltre a questo metodo di navigazione strutturato, però, è necessario anche avere un meccanismo di navigazione "trasversale" dei documenti attraverso riferimenti esterni. Infatti è un caso comune che all'interno di un documento si trovino riferimenti ad altri documenti contenuti nella base di conoscenza, e a questi ultimi deve essere possibile l'accesso, sempre attraverso un collegamento ipertestuale.

Capitolo 3: Soluzioni esistenti

In questo capitolo verrà fatta una panoramica sulle tipologie di servizi on line offerti dalla Pubblica Amministrazione. Successivamente verranno le esperienze più significative maturate nell'ambito dei servizi interattivi, con particolare attenzione agli Sportelli Unici per le Attività Produttive.

Lo stato dei servizi interattivi in Italia, secondo i dati riportati in [Ambriola02], si trova ancora in una fase decisamente acerba. Secondo la ricerca Formez [Bevilacqua02] utilizzata da questa pubblicazione, nonostante circa la metà dei comuni italiani sia dotata di un sito web istituzionale (in particolare per quel che riguarda le grandi città), i servizi offerti consistono, per la maggior parte, nella fornitura di informazioni. Inoltre sembra che tra le possibilità offerte da una presenza in Internet, sia stata recepita soprattutto quella di poter presentare al cittadino un equivalente elettronico dei documenti cartacei già esistenti. In tabella 1, dove sono riportati i risultati di un'indagine Formez, si nota chiaramente che, per quello che riguarda i servizi forniti direttamente dalla pubblica amministrazione (escludendo quindi gli spazi autogestiti da associazioni ed enti), la percentuale di servizi interattivi è largamente sproporzionata rispetto alla semplice pubblicazione di materiale. Il servizio interattivo più diffuso nei siti istituzionali, infatti, è quello del forum con i decisori politici e con l'amministrazione, realizzato nel 12,3% dei comuni presi in esame; di converso, se si considerano i servizi consistenti nella semplice pubblicazione di informazione al cittadino, si passa dalla discreta diffusione degli atti del governo locale (presenti sui siti istituzionali del 44,5% dei comuni considerati) alla notevole presenza di informazioni sul comune (presenti sui siti istituzionali del 79,7% dei comuni considerati). Sempre secondo i dati riportati dall'indagine Formez e mostrati in tabella 2, anche i siti web delle province presentano informazioni principalmente di natura statica, con uno

stacco ancora più netto rispetto ai dati forniti per i comuni.

Servizi	attivi
Pagamento di servizi	6,30%
Prenotazione di servizi	6,30%
Rilascio on-line di cert.	7,60%
Accettazione on-line moduli	9,80%
Forum amministrazione	12,30%
Spazi autogestiti da enti	20,90%
Link ad altri siti della P.A.	42,40%
Atti del governo locale	44,50%
Normativa generale	46,10%
Modulistica	49,40%
Info concorsi e appalti	64,60%
Info città	77,60%
Info comune	79,70%

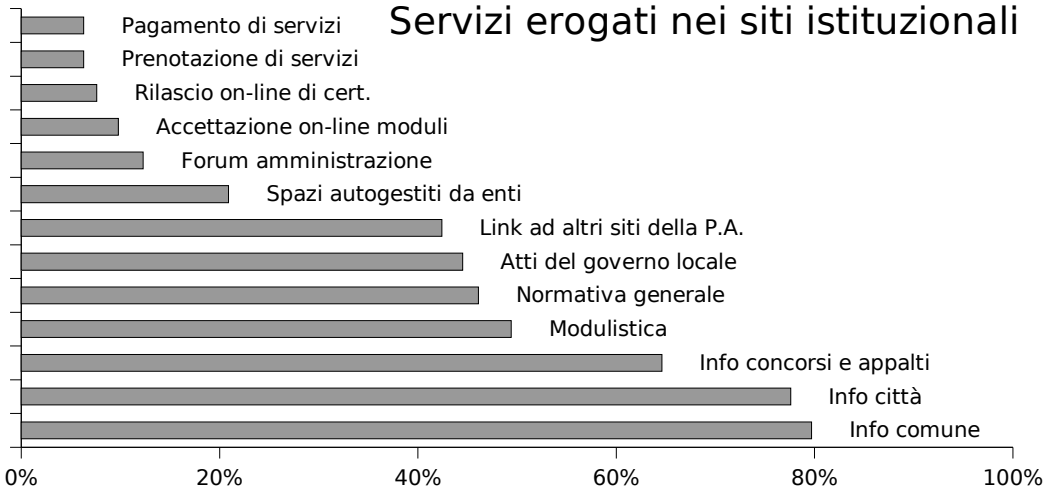

Tabella 1: servizi istituzionali

cfr. [Ambriola02]

Servizi	attivi
Rilascio on-line di cert.	6,80%
Accettazione on-line moduli	8,50%
Prenotazione di servizi	10,20%
Forum amministrazione	35,60%
Spazi autogestiti da enti	47,50%
Atti del governo locale	72,90%
Normativa generale	76,30%
Modulistica	84,70%
Link ad altri siti della P.A.	86,40%
Info provincia (eventi)	94,90%
Info provincia (servizi)	96,60%
Info concorsi	98,30%

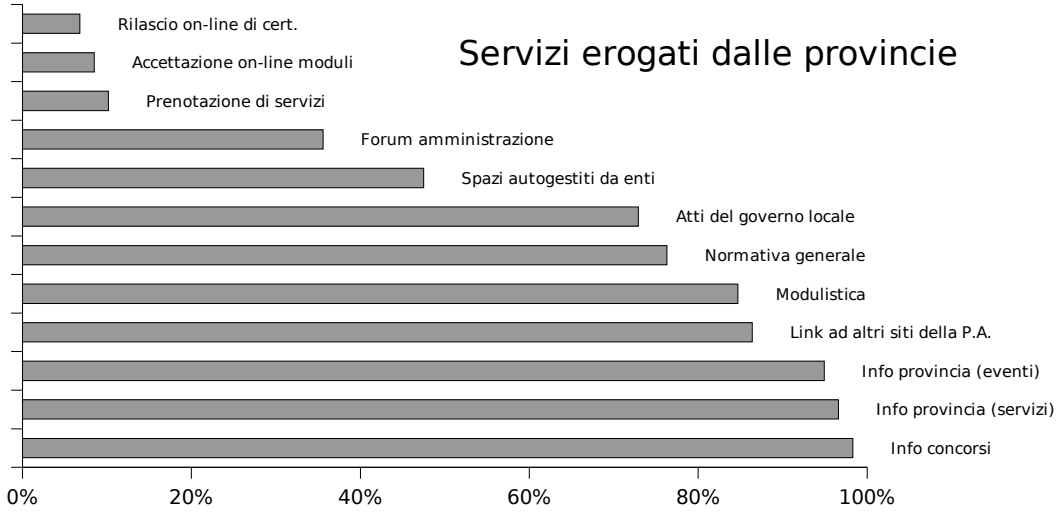

Tabella 2: servizi provinciali

cfr. [Ambriola02]

Nonostante la scarsa diffusione attuale dei servizi interattivi all'interno dei siti istituzionali, si prevede che la crescente alfabetizzazione informatica dei cittadini e l'esempio positivo delle amministrazioni che hanno supportato la realizzazione di servizi evoluti per i loro siti porti ad una crescita di questo tipo di servizi; anche perché, secondo i dati riportati in [Bevilacqua02], il problema principale nell'adozione delle nuove tecnologie informatiche all'interno dei comuni non consiste principalmente in un "conservatorismo" della pubblica amministrazione che limita la spinta alle innovazioni, ma piuttosto nella scarsa preparazione in materia informatica di tecnici e personale amministrativo.

La situazione illustrata nei paragrafi precedenti è riassunta nella tabella 3, che presenta la percentuale di servizi alle imprese e ai cittadini erogati attraverso siti istituzionali secondo un'indagine Rur, Assinform e Censis dell'anno 2000 riportata su [Ambriola02]. Un dato interessante da notare è che nessuno dei comuni minori, e solo il 2,1% dei comuni capoluogo presenta (anche solo in via sperimentale) un servizio interattivo per effettuare transazioni relative allo Sportello Unico per le Attività produttive. Nonostante, quindi, l'importanza e i vantaggi di un servizio interattivo che gestisca le transazioni del SUAP, la situazione fotografata in Italia è ancora ad uno stato pionieristico.

Servizi alle imprese	**Regioni**	**Province**	**Comuni capoluogo**	**Altri Comuni**
Informazioni dettagliate sullo svolgimento di gare (spesso mediante download di documenti)	59,1	69,06	62,06	37,6
Implementazione modello e-procurement (possibile risposta al bando)	4,5	9,8	27,1	5,5
Scaricamento moduli Sportello Unico Attività Produttive	4,5	9,8	27,1	5,5
Transazioni (almeno sperimentali) con Sportello Unico Attività Produttive	-	-	2,1	-
Servizi ai cittadini				
Indicazioni per la prenotazione ai servizi pubblici	-	2,2	5,2	0,6
Prenotazione a distanza dei servizi pubblici	4,5	-	2,1	-
Presenza di sistemi per pagare beni o servizi	4,5	-	2,1	-
Presenza di dispositivo SSL per transazioni commerciali sicure	9,1	9,8	4,2	2,5

Tabella 3: servizi alle imprese

I servizi on line

La vasta tipologia dei servizi on line forniti dalla pubblica amministrazione necessita ovviamente di un qualche tipo di classificazione. Di seguito riporteremo alcuni dei modelli di classificazione proposti per catalogare i tipi di servizi on line. Naturalmente questi modelli sono in larga parte ortogonali tra loro, ed è plausibile che uno stesso servizio possa essere definito specificando contemporaneamente l'appartenenza a classi di modelli differenti.

Classificazione per eventi della vita

Il primo avviso di e-government, diffuso dal dipartimento per l'innovazione e le tecnologie nel 2002, presenta in allegato [ConsMin02] una classificazione dei servizi relativa agli “eventi della vita”, basata su analoghe classificazioni a livello europeo. Il modello di riferimento proposto prevede l'utilizzo di una “metafora di comunicazione” per classificare i servizi offerti non in base all'organizzazione tradizionale della pubblica amministrazione, ma piuttosto in base alle reali necessità della vita del cittadino, che potrà orientarsi meglio in un modello di classificazione più vicino alla sua esperienza.

Queste necessità sono state raggruppate in due macrocategorie, relative al soggetto dell'evento: i servizi ai cittadini e i servizi alle imprese[4].

Per i cittadini sono definite 15 categorie di eventi:

1. Essere cittadino
2. Avere un figlio
3. Avere una famiglia
4. Vivere in salute
5. Abitare
6. Studiare

4 Alla categoria “imprese” appartengono anche i professionisti, le fondazioni e le associazioni

7. Lavorare
8. Percepire la pensione
9. Pagare le tasse
10. Fare e subire una denuncia
11. Usare un mezzo di trasporto
12. Vivere il tempo libero e la cultura
13. Fare sport
14. Andare all'estero
15. Vivere l'ambiente

Per le imprese sono definite dodici categorie di eventi:

1. Aprire una nuova attività
2. Modificare un'attività
3. Sviluppare un'attività
4. Terminare un'attività
5. Finanziare un'attività
6. Gestire il personale
7. Possiede immobili
8. Pagare le tasse
9. Registrare marchi e brevetti
10. portare ed esportare
11. Fare e subire una denuncia
12. Salvaguardare l'ambiente

Ogni servizio fornito dalla pubblica amministrazione riguarda uno di questi eventi, anche se non è escluso che alcuni servizi riguardanti categorie differenti possano venire forniti attraverso un'unica interfaccia. Un esempio può essere anche una realizzazione di SUAP, in cui i servizi di apertura, modifica o terminazione di un'attività sono presentati attraverso una singola applicazione.

Classificazione per area applicativa

[Ambriola02] Presenta un tipo di classificazione che si basa sull'oggetto e la finalità del servizio offerto, e propone un esempio di possibili aree applicative che riportiamo di seguito.

- Ambiente e territorio
- Arte e cultura
- Economia
- Formazione
- Giustizia
- Innovazione
- Lavoro e politiche sociali
- Protezione civile
- Salute
- Stato
- Tutela della privacy

Classificazione per tipologie di interazione

Un'ulteriore classificazione, proposta anch'essa in [Ambriola02], distingue i vari servizi a seconda della modalità di fruizione da parte dell'utente. Il principale vantaggio di una classificazione di questo genere è che servizi funzionanti secondo una stessa modalità possono essere progettati seguendo le medesime linee guida e realizzati utilizzando le medesime tecnologie. In questo modo, una volta stabilito il tipo di servizio da realizzare, può essere valutato l'utilizzo delle esperienze di progettazione e della tecnologia usata per i servizi presenti nella stessa tipologia.

La classificazione proposta nell'opera citata divide i servizi secondo tre diverse tipologie:

- servizi informativi:
  questo tipo di servizi consiste essenzialmente nel presentare delle

informazioni on line, sia attraverso la presentazione diretta all'interno della pagina web, sia attraverso documenti scaricabili dall'utente;

- servizi di comunicazione:
  questo tipo di servizi consiste nel fornire all'utente la possibilità di comunicare con altri utenti, con i responsabili dell'ente, con gli amministratori del servizio o con altri soggetti attraverso una comunicazione personale (ad esempio con l'utilizzo di posta elettronica) o pubblica (come può essere quella di un forum civico);
- servizi transattivi:
  questo tipo di servizi consente di operare transazioni complesse tra l'ente che fornisce il servizio e il cittadino. Esempi pratici di questa tipologia di servizi possono essere l'esecuzione di una procedura amministrativa direttamente on line, il pagamento delle tasse attraverso carta di credito o un sistema di prenotazioni per eventi di interesse pubblico.

Oltre a questa classificazione generale, in [Ambriola02] è proposta anche una ulteriore macroclassificazione per i servizi interattivi on line:

- interazione “a una via” (one-way).
  I servizi di questo tipo sono monodirezionali. In pratica l'unica interazione che avviene è il trasferimento di informazione da parte dell'ente verso il cittadino. In questa categoria si collocano i servizi di tipo informativo;
- interazione “a due vie” (two-way).
  I servizi di questo tipo implicano uno scambio di informazioni bidirezionale tra ente e cittadino, che può fornire direttamente on line le informazioni necessarie all'erogazione del servizio. In questa categoria sono collocati i servizi di comunicazione e transattivi.

Esempi di servizi interattivi:

Servizi culturali e turistici

Il Ministero per i beni e le attività culturali[5] ha realizzato un portale che offre diversi servizi all'utente. Questi servizi sono sia di tipo informativo che di tipo transattivo. Tra i servizi informativi vi è la possibilità di conoscere l'ubicazione degli archivi di stato e i documenti in essi contenuti, la possibilità di effettuare ricerche sul sistema bibliotecario nazionale e una sezione dedicata alla multimedialità che contiene informazioni sui beni culturali in Italia, con sezioni specifiche dedicate ad alcuni dei più importanti di essi, come il Colosseo, Pompei, gli Uffizi o il Vittoriano. I servizi transattivi, invece offrono la prenotazione on line ai luoghi d'arte convenzionati con il Ministero, come ad esempio il Museo Egizio di Torino, Villa Pisani a Venezia, la Torre di Pisa, il Colosseo, o il Palazzo Reale a Napoli. Questi servizi di prenotazione, però, non vengono gestiti internamente al sito del Ministero, ma puntano ai portali delle società che si occupano della vendita dei biglietti. Inoltre è presente un'area che consente l'acquisto on line di pubblicazioni degli archivi di stato, ma attualmente è in fase di avviamento.

Sistema del lavoro

Il Ministero del Lavoro, per andare incontro sia alle necessità delle aziende di trovare nuovo personale, sia a quelle dei lavoratori, disoccupati e non, di conoscere le offerte disponibili sul mercato del lavoro, ha realizzato un sito web, denominato e-labor[6], che fornisce un servizio di comunicazione tra i due mondi. Una volta registratosi, l'utente può inserire il suo curriculum vitae, consultare le proposte di lavoro delle aziende e consultare i curricula vitae degli utenti iscritti.

5 Http://www.beniculturali.it
6 Https://e-labor.minlavoro.it

I dati anagrafici degli utenti, sono nascosti alla visualizzazione per garantire il diritto alla privacy. Purtroppo, la compatibilità del servizio è garantita solo con alcuni browser web (Internet Explorer e Netscape). Altri tipi di browser (come ad esempio Mozilla e Konqueror) non sono riusciti ad accedere ai servizi offerti.

Agenzia delle entrate

Il Ministero dell'Economia e delle Finanze, per semplificare le operazioni relative all'imposizione tributaria, ha realizzato un sito web[7], che offre diversi servizi transattivi, tra i quali:

- ***Presentazione delle dichiarazioni***

  Questa sezione permette ai cittadini, la compilazione e trasmissione telematica delle seguenti dichiarazioni:

  - Unico persone fisiche;
  - Unico società di capitali;
  - Unico Società di Persone;
  - Unico Enti Non Commerciali;
  - IVA;
  - 770 ordinario;
  - 770 semplificato.

- ***Servizio F24 online***

  Il servizio "F24 online" permette agli utenti di compilare in modo assistito il modello F24 e di effettuare il pagamento on line delle imposte senza doversi recare presso gli sportelli bancari. L'utente, secondo le informazioni presenti sul sito, ottiene dal pagamento on line i seguenti benefici:

  - poter effettuare il pagamento anticipatamente ed avere comunque l’addebito in conto della somma a saldo del modello F 24 alla data di versamento indicata;
  - ricevere dall’amministrazione finanziaria:
    - in via telematica, la segnalazione della correttezza e/o

7 Http://www.agenziaentrate.it

anomalia della richiesta di pagamento inoltrata e, in caso di saldo del modello F24 da addebitare in conto, l'esito dell'addebito comunicato dalla banca che detiene il conto corrente;

- tramite Postel, la quietanza (F 24) del pagamento effettuato.

- ***Emersione del lavoro irregolare***

  Questo servizio consente di compilare la dichiarazione per l'emersione del lavoro irregolare.

- ***Contratti di locazione***

  Questo servizio consente la registrazione on line dei contratti di locazione da parte dei cittadini. Nel sito è riportato anche che per i cosiddetti "grandi contribuenti", che possiedono almeno 100 immobili, la registrazione on line è obbligatoria. Infine la registrazione on line dei contratti di locazione permette il pagamento per via telematica di imposte di registro, bollo ed eventuali interessi e sanzioni, attraverso il modello F24.

- ***cassetto fiscale***:

  Questa sezione[8] consente ai cittadini di accedere in modo sicuro alle proprie informazioni fiscali per visualizzare informazioni quali:

  - dichiarazioni presentate;
  - rimborsi di imposte dirette;
  - versamenti (Modello F24 e F23);
  - dati patrimoniali (atti del registro);
  - codice fiscale, dati anagrafici e residenza;
  - denominazione, partita Iva, domicilio fiscale, sede legale e descrizione dell'attività di una ditta individuale.

Per accedere ai servizi transattivi forniti dal portale, è necessario scaricare dei software specifici, forniti gratuitamente e disponibili per piattaforma Windows e Macintosh, che permettono di compilare

8 A differenza degli altri, questo è un servizio informativo

off line il modulo richiesto e di salvare i dati inseriti in un file che può essere inviato al ministero attraverso il portale stesso. Al termine dell'operazione viene inviata una ricevuta all'utente per comunicare l'avvenuta ricezione del modulo.

Quali servizi offrono regioni e comuni per l'accesso ai procedimenti amministrativi ?

Presentiamo adesso alcuni dei servizi offerti dalle regioni e dai comuni per consentire ai cittadini e alle aziende di accedere ai procedimenti amministrativi[9].

Piemonte

L'Ufficio Relazioni con il Pubblico (URP) della Regione Piemonte[10] offre:

- informazioni sulle attività svolte dagli uffici regionali, attraverso la compilazione di una form nella quale si possono selezionare l'URP destinatario, l'argomento e il contenuto della domanda.
- Collegamento alla pagina della Regione nella quale sono pubblicati i bandi attualmente in corso e per ognuno dei quali è disponibile il testo del documento amministrativo che lo riguarda, assieme al modulo da stampare e compilare per potervi partecipare.
- Possibilità di download dei moduli per l'accesso ai documenti amministrativi.
- Possibilità di prenotare on line una postazione multimediale presso uno dei vari URP regionali per navigare gratuitamente nel sito della Regione.
- Download dei moduli per l'autocertificazione o della dichiarazione sostitutiva di notorietà.

Emilia Romagna

Dal sito web dell'Ufficio Relazioni con il Pubblico della Regione Emilia Romagna[11] è possibile accedere all'area dei servizi on line, che comprendono un'area dedicata alla “Modulistica in rete”[12]. L'obiettivo di questa sezione è quello di consentire al cittadino sia di accedere in modo più semplice alle varie iniziative regionali, sia di snellire l'iter burocratico. Queste iniziative sono suddivise tra quelle

9 La selezione delle Regioni e dei Comuni è avvenuta basandoci su [Ambriola02]
10 http://www.regione.piemonte.it/governo/urp/index.htm
11 http://www.regione.emilia-romagna.it/urp/
12 http://www.regione.emilia-romagna.it/fr_modulistica.htm

che hanno una scadenza temporale e quelle per cui invece non esiste un limite di tempo per presentare la domanda. Inoltre, per ogni iniziativa è possibile accedere a pagine contenenti una breve descrizione dell'iniziativa stessa; i moduli stampabili che si devono presentare per avviare la relativa procedura di attivazione e il documento amministrativo dal quale deriva l'iniziativa.

Lazio

La regione Lazio, similmente all'Emilia Romagna, ha scelto di rendere disponibili i moduli per l'accesso ai propri servizi passando attraverso l'area dell'Ufficio Relazioni con il Pubblico[13]. Tuttavia la modulistica è organizzata secondo una classificazione differente: i moduli sono infatti divisi in base alla loro categoria di appartenenza (come ad esempio agricoltura, lavori pubblici e turismo). Selezionando il singolo modulo ne viene visualizzata una versione digitale che può essere scaricata dall'utente.

Campania

Anche qui, la regione offre al cittadino la possibilità di scaricare on line i moduli ma il servizio è limitato a quello per il reclamo e a quello generico per l'accesso agli atti amministrativi.

Torino

Il Comune di Torino[14] sta lavorando su due fronti nell'ambito dei servizi interattivi che mette a disposizione dei cittadini e delle imprese: da una parte ha lanciato tre iniziative chiamate Torino Facile, Torino Facilissima e Torino In Diretta, che consentono l'utilizzo di particolari carte elettroniche; dall'altra ha realizzato il sito “Sportello Web”[15] che è un portale strutturato secondo schemi tradizionali.

Torino Facile è una carta elettronica, dotata di un microchip che attualmente è disponibile soltanto per i dipendenti comunali e per i

13 http://213.175.14.104/produzione/urp/sitourp.nsf/home
14 http://www.comune.torino.it/index.htm
15 http://www.comune.torino.it/sportelloweb/index.htm

professionisti. Questa carta può essere utilizzata sia quando l'utente è on line sia quando questi è off line. Quando viene impiegata nella prima modalità, certifica l'identità dell'utente consentendogli così di eseguire le transazioni che richiedono una certificazione o richiedono di firmare elettronicamente i documenti. Quando invece viene impiegata in modalità off line, grazie al fatto che mantiene in memoria i dati dell'utente, gli permette di accedere ai servizi e alle iniziative lanciate dalle società pubbliche e private che partecipano a questo progetto. Torino Facilissima è invece una versione ridotta della carta elettronica vista prima, che non è dotata di un microchip e il cui obiettivo è "L'implementazione di un sistema di autenticazione per l'accesso ai servizi telematici del Comune di Torino in grado di essere diffuso su larga scala, a tutti i cittadini residenti e non che ne faranno richiesta"[16]. Nel sito, viene inoltre specificato che la carta è in grado di emettere "Certificati di autenticazione digitale che non richiedono di essere memorizzati su carte a microprocessore, ma utilizzati nei normali browser per la navigazione Internet". Attualmente i servizi interattivi di cui può usufruire il cittadino che possiede Torino Facilissima sono: l'autocertificazione; il calcolo e il pagamento dell'ICI; la dichiarazione del cambio di abitazione (in fase di rilascio); il pagamento on line di TARSU, COSAP e CIMP; e le visure tributarie[17]. Infine l'iniziativa Torino In Diretta, lanciata in contemporanea con Torino Facilissima, offre al cittadino la possibilità di collegarsi ad Internet (tramite modem analogico o ISDN) e di avere una e-mail, accessibile sia da un tradizionale client di posta elettronica sia dal web, pagando il costo di una normale telefonata urbana. Lo Sportello Web offre la possibilità di accedere a diversi servizi, sia transattivi che informativi. Oltre a fornire servizi disponibili solo per i possessori di carta Torino Facile/Facilissima, il portale consente di

16 http://www.torinofacile.it/informati/tofacilissima.htm
17 Ripreso da: http://www.torinofacile.it/registrati/index.htm

richiedere:

- certificati di Stato Civile (nascita,matrimonio, morte).
- Autocertificazione telematica assistita.
- Prenotazione delle certificazioni o delle attestazioni urbanistiche.
- Richieste di copie di provvedimenti amministrativi edilizi.

I dati relativi alle richieste vengono compilati attraverso dei form presenti sulla pagina web, e viene data la possibilità al cittadino di andare a ritirare il certificato richiesto allo sportello o di riceverlo a casa via posta normale.

Un ulteriore servizio transattivo offerto dal portale consente di pagare le multe direttamente on line attraverso una transazione sicura con carta di credito.

Per quanto riguarda i servizi informativi offerti, il cittadino ha la possibilità di:

- effettuare una ricerca sulle delibere;
- accedere alla banca dati delle ordinanze;
- effettuare delle ricerche tra gli atti e i filmati delle sedute del consiglio comunale;
- accedere alle determinazioni dirigenziali;
- esaminare la banca dati degli eventi culturali a Torino;
- consultare il Piano Regolatore Generale.

Infine, per quanto riguarda i servizi di comunicazione, è possibile:

- inviare dei messaggi ai responsabili degli uffici comunali attraverso un'apposita interfaccia web;
- chiedere di ricevere tramite e-mail o SMS informazioni di interesse collettivo, come eventi culturali, cambi nella viabilità o comunicati stampa.

Milano

Il Comune di Milano, suddivide i servizi on line accessibili dal proprio sito web[18] tra quelli rivolti al cittadino e quelli rivolti alle imprese. Nella prima categoria troviamo:

18 http://www.comune.milano.it/

- il calcolo dell'ICI;
- la richiesta di certificati (come quello di nascita, residenza, matrimonio, etc.[19]) che i residenti a Milano possono ricevere a casa pagando in contrassegno[20];
- il controllo della richiesta del cambio di residenza;
- la possibilità di chiedere all'ufficio Oggetti Smarriti se un oggetto è stato ritrovato;
- controllare quando è stata stipulata la concessione per una celletta ossario e cinerario[21] in uno dei quattro cimiteri e quando si debba effettuarne il rinnovo.

Nella categoria indirizzata alle imprese, invece, troviamo:

- calcolo dell'ICI;
- calcolo del TARSU;
- accesso allo Sportello Unico per le Attività Produttive[22].
  Utilizzando quest'ultimo servizio, l'utente può navigare tra le varie categorie, a loro volta suddivise in sotto categorie, finché non trova il modulo relativo alla Domanda Unica da compilare insieme agli eventuali allegati che devono essere presentati insieme al documento.

Bologna

Il Comune di Bologna, che partecipa al progetto DALI[23] (il cui obiettivo è quello di "migliorare l'interazione tra i cittadini e l'amministrazione comunale attraverso servizi e prodotti multimediali informatici"), offre principalmente quattro aree di servizi, sia transattivi che informativi:

- la prima area è quella relativa all'ICI[24] (Imposta Comunale sugli Immobili), dove il cittadino ha la possibilità di accedere a tutte le

19 http://www.comune.milano.it/certificati/index.html
20 I cittadini residenti al di fuori del Comune sono costretti ad utilizzare il telefono e pagare la transazione attraverso carta di credito per ricevere i certificati a domicilio
21 http://www.comune.milano.it/servizifunebri/concessioni.html
22 http://web3.comune.milano.it/onlyone/SuInformazioni/informa.nsf/
23 http://servizi.comune.bologna.it/eu/dali/daligen.html
24 http://a9.comune.bologna.it/portaleici/index.html

informazioni necessarie per sapere cosa sia questa tassa, come calcolare l'importo che deve pagare, può scaricare il modulo di pagamento, può consultare gli orari di apertura degli uffici in cui può effettuare il pagamento oppure, previa registrazione, pagare la tassa on line attraverso carta di credito.

- Nell'area relativa alla Ricerca Pratiche[25], è possibile visualizzare un atto protocollato fornendo l'anno e il numero di registrazione come parametri di ricerca, oppure il cittadino può venire a conoscenza dell'ufficio comunale in cui si trovi un documento che ha presentato, insieme agli eventuali procedimenti amministrativi a cui può avere dato origine. In quest'ultimo caso, è inoltre possibile conoscere i termini temporali entro i quali devono essere conclusi i vari procedimenti, il suo status nell'iter burocratico e il funzionario comunale incaricato di seguirlo.
- L'area, denominata “Sportello per Edilizia e Imprese”[26] e che è stata sviluppata all'interno del progetto SUPER[27] (un progetto europeo con il fine di “rimuovere gli ostacoli che impediscono la completa realizzazione di un Singolo Punto di Accesso per le imprese nella Pubblica Amministrazione”), permette di conoscere gli eventuali aggiornamenti normativi, di scaricare le Domande Uniche e di firmarle elettronicamente per mandarle on line.
- Infine l'ultima area, denominata CupWEB[28], consente al cittadino di prenotare via web i servizi medici che le strutture sanitarie pubblicate e private, convenzionate con questo servizio, mettono a disposizione.

25 http://www.comune.bologna.it/servizionline/pratiche/ricercapratiche.php
26 http://sportelloediliziaimprese.comune.bologna.it/SportelloUnico/Informa.nsf
27 http://www.superproject.org/en/index1.htm
28 http://www.cup2000.it/cup2000/it/domestica/welcomepage.asp

Roma

Tra i servizi on line messi a disposizione dal Comune di Roma ai cittadini e alle imprese troviamo:

- la notifica dei bandi e dei concorsi comunali attualmente in vigore[29];
- il calcolo dell'ICI;
- l'accesso on line alle delibere comunali;
- la possibilità di pagare on line ICI e TARSU (previa registrazione) e di verificare se lo stato dei pagamenti;
- l'accesso allo Sportello Unico per le Imprese[30]. In questa sezione, previa registrazione, l'utente può navigare nella base documentale della normativa comunale, scaricare il modulo per la Domanda Unica e controllare lo stato di avanzamento della propria pratica.

Pesaro

Il servizio "Progetto Scuola"[31], realizzato dal Comune di Pesaro, si propone di offrire alle famiglie "un contatto diretto e continuo con la scuola permettendo così una partecipazione più attiva al percorso scolastico dei propri figli" che potranno controllare il numero delle assenze, compilare on line le giustificazioni, prendere visione dei documenti o delle comunicazioni inviate dalla scuola e controllare le valutazioni via web.

Un altro progetto realizzato dal Comune di Pesaro per i cittadini è quello soprannominato "Agenda del Cittadino"[32]. Questo servizio consente di ricevere tramite SMS o sulla casella di posta elettronica indicata al momento della registrazione, notifiche o avvisi come eventuali disservizi di gas/acqua, scadenze dei pagamenti o eventi culturali e istituzionali della città di Pesaro. Il Comune ha infine reso possibile il pagamento via web delle multe[33].

29 http://www.comune.roma.it/bandieconcorsi/
30 http://www.suroma.comune.roma.it/home.asp
31 http://www.comune.pesaro.ps.it/progettoscuola/proscuola.asp
32 http://www.comune.pesaro.ps.it/asp/agendacittadino/
33 http://www.comune.pesaro.ps.it/pagamentomulte/

Tra i servizi on line per le imprese, invece, troviamo:

- la possibilità per i fornitori del comune di consultare on line lo stato del mandato di pagamento delle loro fatture[34];
- l'accesso alla banca dati delle concessioni edilizie con la possibilità compilare una nuova domanda o di controllare lo status burocratico di una domanda in corso di esame[35];
- il “Settore delle Attività Economiche”, presso il quale è possibile scaricare il modulo prestampato di una Domanda Unica, conoscere i riferimenti normativi, i requisiti e gli allegati da presentare insieme al suddetto documento, conoscere lo stato di una pratica e navigare nella base documentale[36]. Sempre in quest'area, c'è una sezione dedicata ai commercianti, che possono scaricare la modulistica e le informazioni necessarie per avviare i saldi o la liquidazione della loro attività.

Panoramica delle tecnologie esistenti riguardo ai SUAP

La realizzazione di un servizio interattivo per la Pubblica Amministrazione, in particolare quella dello Sportello Unico per le Attività Produttive (SUAP), in genere richiede l'utilizzo di una vasta gamma di tecnologie differenti, come sistemi di Knowledge Management, Content Management, Web Publishing e gestione della sicurezza. Le soluzioni proposte fino a questo momento sono state costruite adattando, trasformando ed estendendo quelle già esistenti, in modo che possano offrire le funzionalità necessarie al SUAP. Questa scelta è stata fatta per diverse ragioni, quali ad esempio:

- La necessità di non abbandonare le infrastrutture informatiche esistenti.
- Il poco training necessario per istruire gli utenti all'utilizzo delle nuove funzionalità.
- Il non dovere cambiare la modalità di utilizzo della struttura già

34 http://www.comune.pesaro.ps.it/fatture/
35 http://www.comune.pesaro.ps.it/asp/ediliziaweb/
36 http://www.comune.pesaro.ps.it/asp/basedocumentale/

esistente.

Ad esempio, uno degli scenari più comuni è quello in cui un software gestionale, ottimizzato per utilizzi specifici di back office, viene esteso ed integrato per pubblicare il contenuto della sua base di dati sul web e per permettere operazioni di front office all'utente esterno. Operare questo tipo di strategia in modo da soddisfare i criteri di usabilità normalmente richiesti dai servizi interattivi, però, è un'operazione estremamente complessa e difficile, considerando che gli applicativi esistenti di solito non sono stati progettati per essere facilmente estendibili e non offrono funzionalità intrinseche per l'interoperabilità con altri elementi software. Questo ha portato a sviluppare soluzioni poco flessibili e scarsamente integrate, a causa dei compromessi necessari per adattarsi ai limiti delle applicazioni preesistenti (che erano state progettate per soddisfare altri requisiti). Inoltre, molto spesso, queste applicazioni di base sono particolarmente pesanti e richiedono delle infrastrutture costose e complesse da gestire.

Vediamo adesso una panoramica sulle tipologie di applicazioni attualmente esistenti per lo Sportello Unico:

1. Gestionali: In questo caso, l'applicazione viene costruita a partire da una base di conoscenza già esistente, generalmente gestita da un database relazionale o gerarchico, aggiungendo la possibilità di accedere alle informazioni attraverso il web e utilizzando documenti in formato proprietario come elementi atomici da inserire all'interno della base di conoscenza. Questo tipo di soluzioni hanno però diversi svantaggi: per prima cosa la conversione in un formato adatto alla pubblicazione su Web dei dati contenuti all'interno del database è solitamente un'operazione complessa, visto che da una parte (la base di dati) si ha un tipo di informazione incentrata sui dati ed estremamente regolare, dall'altra (la pagina Web) si ha un tipo di informazione incentrato sul contenuto e con una struttura altamente irregolare. Inoltre, il

fatto che il documento venga considerato dal sistema come un unico blocco binario non permette di estrapolare frammenti di informazioni in maniera automatica. Infine, in questo tipo di soluzioni il mantenimento della base di conoscenza viene demandato ad applicazioni specifiche, che richiedono di essere installate, configurate e mantenute su tutte le macchine client, rendendo impossibile l'aggiornamento della base di conoscenza da postazioni che non siano state preventivamente predisposte a questo scopo.

2. Web-centric: In questo caso il servizio è fornito attraverso un certo numero di pagine HTML statiche, ognuna creata indipendentemente dall'altra, che accedono ad una base di conoscenza, anch'essa statica, composta da documenti in formato binario (ad esempio Doc o PDF) memorizzati sul filesystem. Questa soluzione è stata adottata nei casi in cui la base di conoscenza normativa era stata creata con una semplice conversione in formato digitale del documento cartaceo e non era stata prevista nessuna infrastruttura informatica per gestirli. Nonostante questa soluzione sia facile da realizzare quando si devono gestire basi di conoscenza con un numero limitato di dati, non appena aumenta la quantità di informazioni amministrate dal sistema, tutta la struttura diventa difficilmente manutenibile, dato che il più piccolo cambiamento strutturale potrebbe dover essere riportato in ognuna delle pagine che compongono il framework, senza la possibilità di propagarlo automaticamente. Inoltre le limitazioni intrinseche sia dello standard HTML, sia del protocollo HTTP non consentono di realizzare funzionalità avanzate. In effetti è difficile che i sistemi di questo tipo riescano a superare il secondo livello di interattività dei servizi.

Un difetto che accomuna le soluzioni esaminate è quello di non poter costruire nuovi documenti in maniera anche parzialmente automatica, a partire dalle informazioni contenute nella base di

conoscenza. I documenti utilizzati, infatti, vengono considerati come una semplice trasposizione digitale del documento cartaceo, senza le informazioni semantiche che ne descrivono la struttura interna. Questo obbliga a considerare il documento come un unico blocco monolitico, da cui non si possono estrarre facilmente informazioni significative e al quale non si possono aggiungere automaticamente informazioni in modo incrementale. Infine, le soluzioni attualmente esistenti non forniscono un adeguato supporto all'inserimento dei dati, come ad esempio una validazione dei dati inseriti o il riempimento automatico dei campi che possono essere calcolati o estratti dalla base di conoscenza.

Capitolo 4: Tecnologie utilizzate

1. *In questo capitolo verranno presentate le tecnologie utilizzate per implementare la nostra soluzione, fornendo per ognuna di esse la motivazione che ha portato alla sua scelta. Queste tecnologie consistono in Java come linguaggio di implementazione, XML come formato di rappresentazione dei documenti, un qualsiasi database XML:DB-compliant come data repository per i documenti, Apache Tomcat come application server e XSL come linguaggio di presentazione.*

Java

Dovendo scegliere un linguaggio di programmazione con cui implementare un framework per la gestione dello sportello unico, il Java è sembrato immediatamente la soluzione migliore: anche se esistono linguaggi che lo superano in efficienza, il linguaggio sviluppato da Sun li batte in robustezza, facilita di programmazione, supporto per il debugging e pulizia di design. Queste ultime caratteristiche sono state ritenute di un'importanza più rilevante rispetto alla velocità di esecuzione, dato che non erano previsti requisiti stringenti sul tempo di risposta del sistema. Inoltre, il fatto che il codice Java sia multipiattaforma lo rende particolarmente indicato per un software rivolto alle pubbliche amministrazioni, dove solitamente non esiste un ambiente standard per quel che riguarda i componenti informatici. Purtroppo quest'ultimo vantaggio è in parte mitigato da incompatibilità tra le diverse versioni delle Java Virtual Machine esistenti, per cui, ad esempio, un programma che gira sotto la versione 1.4.2 della JVM Sun, potrebbe non funzionare sotto la JVM 1.3.1 di IBM, o viceversa. Durante la fase di sviluppo, quindi, si è posta una particolare attenzione a testare il codice su diverse versioni del Runtime Environment in modo da essere sicuri che il framework fosse realmente portabile. Per maggiori approfondimenti su Java, si può consultare [JoySteeleGoslingBracha00], [ArnoldGoslingHolmes00] e, naturalmente, il sito ufficiale di Sun[37].

37http://java.sun.com/

XML

I linguaggi di markup per i sistemi informatici hanno una lunga storia, che inizia dai primi anni 70 con il GML: un sistema ideato presso i laboratori IBM nell'ambito di un progetto di ricerca sui sistemi informativi integrati per uffici legali. In seguito questo lavoro venne utilizzato come base per il SGML (Standard Generalized Markup Language), un linguaggio di markup standard ANSI che avrebbe in seguito ottenuto un'importanza strategica in numerosi progetti. Purtroppo la complessità di SGML lo rendeva pesante ed inutilmente farraginoso rispetto ai problemi di media difficoltà. Per risolvere questa situazione il W3C (il consorzio che si occupa degli standard per il Web) decise di crearne una versione semplificata da proporre come il nuovo linguaggio per la pubblicazione elettronica al posto di HTML[38].

I vantaggi di XML per la gestione documentale sono diversi, e di seguito se ne elencano i più significativi:

1. Standard Aperto: le specifiche di XML sono disponibili a chiunque voglia scrivere del software che lo utilizza. In questo modo il linguaggio non è legato alle sorti economiche o alle scelte strategiche di una particolare azienda che ne detiene i diritti e che potrebbe cessare di supportarlo arbitrariamente, come invece avviene alle volte per i formati proprietari.
2. Estremamente supportato: Al momento sono disponibili un gran numero di librerie per la gestione di XML su una vastissima gamma di linguaggi, da quelli di uso più comune, come Java o C/C++, ai linguaggi di scripting, come PHP Python o Perl, fino a quelli utilizzati in settori specializzati come ML, Lisp o RPG.
3. Informazioni semantiche: XML permette, attraverso il meccanismo dei markup, di dare una struttura al documento, indicando il significato dell'informazione contenuta al suo interno.

38 Per maggiori dettagli sulla storia di XML,si può consultare il sito del W3C (http://www.w3.org/XML/hist2002)

4. Validazione: Ad un documento XML, solitamente, vengono associati una DTD o un XML-Schema, che definiscono la struttura semantica del documento stesso e che rendono possibile validarlo. Nonostante l'XML-Schema permetta una flessibilità maggiore, le DTD continuano a venire utilizzate a causa di una maggiore semplicità di definizione

Quale modello per la gestione dei flussi XML ?

Dato che il nostro framework deve gestire dei documenti XML, una delle prime scelte architetturali che abbiamo dovuto affrontare è stata quella di decidere il modello (o i modelli) di gestione dei flussi XML da adottare. Attualmente esistono cinque modelli che hanno una certa predominanza e per cui sono stati realizzati strumenti di sviluppo abbastanza maturi, descritti in [Obasanjo01]. Dopo una breve panoramica e la presentazione di alcune implementazioni, illustreremo le motivazioni che ci hanno portato alle nostre scelte di progetto.

Modello Push

In questo modello l'elemento attivo del sistema è il parser. Quest'ultimo, infatti, invia un particolare evento all'applicazione ogni volta che individua un elemento del documento. L'esempio più noto di realizzazione di questo modello sono le API SAX, in cui il parser invia delle chiamate a determinati metodi dell'applicazione ogni volta che individua un elemento nel flusso XML. Il vantaggio principale del modello Push è che, quando viene elaborato un documento XML, questi non viene caricato interamente in memoria, ma vengono processate solo le informazioni relative al nodo attuale. Questo rende possibile l'elaborazione di documenti XML di grandi dimensioni, che potrebbero anche superare le centinaia di gigabyte, senza dovere utilizzare la memoria interna per processarli. Lo svantaggio principale di questo modello è che nel suo utilizzo, ovviamente, non vengono mantenute automaticamente certe informazioni di contesto

o di stato (come le relazioni padre-figlio o la profondità di un nodo nell'albero XML). La gestione di queste informazioni sarà quindi a carico dello sviluppatore. Un altro problema minore, anche se non di natura tecnica, di questo modello è che, lavorando sul flusso invece che sulla rappresentazione ad albero, lo sviluppo di un'applicazione basata su SAX risulta poco intuitivo e il modello stesso richiede dei tempi più lunghi nell'apprendimento.

Modello Pull

Durante l'elaborazione di un documento XML con un parser basato sul modello Pull, è l'applicazione a controllare il flusso del programma, richiedendo esplicitamente l'invio di un evento al parser quando lo ritiene necessario, invece di aspettare che gli siano inviati. Similmente al modello Push, i parser basati su questo modello possono scorrere il flusso soltanto in avanti, e possono mostrare le informazioni di un nodo soltanto nel momento in cui lo analizzano. Questo modello offre quindi la stessa efficienza nella gestione della memoria del modello Push, ma ha il vantaggio di utilizzare un modello di programmazione più familiare al programmatore medio[39]. Gli esempi più noti di questo modello sono XmlReader class del .NET Framework e Common API di XML Pull Parsing.

Modello ad albero

Il modello ad albero, è un modello ad oggetti che rappresenta un documento XML attraverso una struttura ad albero. Questo modello è composto da oggetti che corrispondono a diversi elementi, definiti dalle specifiche pubblicate dal W3C, che si possono trovare in un documento XML, come elementi, attributi, istruzioni di processing e commenti. Attraverso una API specifica, l'applicazione è in grado di caricare, memorizzare, accedere, interrogare, modificare e cancellare elementi da un documento XML. L'esempio principale di

39 Negli ultimi tempi, però, il modello di programmazione a eventi, grazie all'uso che ne fanno diversi sistemi grafici e interattivi, sembra aver raggiunto un buon livello di diffusione.

una API basata sul modello di rappresentazione ad albero è il Document Object Model (DOM) del W3C, di cui sono state realizzate diverse implementazioni. Due delle più diffuse sono Xerces[40] della Apache Foundation e GNU Jaxp della Free Software Foundation[41]. Tipicamente le API basate sul modello ad albero, caricano l'intero documento in memoria e ciò non limita gli utenti a potere accedere soltanto all'elemento esaminato in qual momento. Lo svantaggio di questo approccio, però, è che non si può impiegare in situazioni in cui si devono gestire documenti XML di grandi dimensioni, dato che occupa una parte significativa della memoria principale.

Modello Cursor

I parser XML cursor rappresentano l'ultima generazione di API progettate per elaborare flussi XML. Un parser di questo tipo utilizza un cursore che si concentra solo su di un nodo alla volta ma, a differenza di quello che accade nei modelli Push o Pull, questo cursore può essere sempre posizionato in qualsiasi parte del documento XML, indipendentemente dalla posizione attuale. Similmente alle API basate sul modello ad albero, i parser XML cursor permettono di navigare, interrogare e manipolare un documento XML caricato in memoria. Però, anche se questo modello non raggiunge la leggerezza dei modelli Push e Pull, potenzialmente potrebbe riuscire ad utilizzare la memoria principale in maniera più efficiente rispetto al modello ad albero. Infatti, al contrario di quest'ultimo, il modello Cursor non associa necessariamente ad ogni token significativo del XML analizzato un oggetto che risieda in memoria. Un'altra differenza significativa tra i due modelli è che, mentre con il modello ad albero per mappare una sorgente di dati esterna è necessario scorrerla tutta e generare i nodi corrispondenti, con il modello Cursor è sufficiente realizzare un'implementazione del cursore direttamente sulla sorgente dati. Ad esempio si potrebbe

40 Per la home page di Xerces per Java: http://xml.apache.org/xerces2-j/index.html
41 http://www.gnu.org/software/classpathx/jaxp/

trattare un insieme arbitrario di oggetti come un documento XML virtuale sul quale eseguire interrogazioni tramite XPath o trasformazioni attraverso fogli di stile XSLT. Due esempi di API basate su questo modello sono XPathNavigator class del .NET Framework e XmlCursorClass di XMLBeans di BEA.

Modello di mapping oggetto/XML

Spesso può essere conveniente trasformare un documento XML in un oggetto del linguaggio utilizzato, in modo da poter gestire i dati in maniera più efficiente e lineare. Il primo vantaggio di questo approccio è che l'occupazione della memoria richiesta può essere drasticamente ridotta, dato che l'informazione non è più memorizzata come nodo o dato esclusivamente testuale, ma piuttosto ha una rappresentazione nativa che certamente è più efficiente in termini di memoria e accessibilità. Ad esempio, un nodo che contenga un'informazione numerica in forma testuale occupa certamente più memoria rispetto alla stessa informazione memorizzata in un campo intero. Inoltre, al momento di effettuare operazioni aritmetiche su quel valore, queste possono essere applicate direttamente senza dover convertire ogni volta la stringa in un valore numerico. Infine, utilizzare un oggetto nativo rispetto ad un'altra rappresentazione XML da il vantaggio di poter utilizzare i costrutti del linguaggio per accedere ai dati del documento senza bisogno di invocare metodi esterni. Il modello, però, presenta anche tutta una serie di limitazioni che non lo rendono la soluzione ottima per tutte le esigenze. Per prima cosa, allo stato attuale delle cose, nessuna delle implementazioni di questo modello permette di rappresentare tutti i tipi di informazione presenti in un documento XML mantenendo la struttura originaria. Ad esempio alcune non sono in grado di gestire le istruzioni di processing e i commenti, ed inoltre non tutti i documenti con contenuti misti riescono ad essere convertiti direttamente in oggetti. Inoltre, l'ordine degli elementi di un documento XML può essere significativo e questa proprietà non

riesce ad essere gestita attraverso una mappatura sugli oggetti, in quanto l'ordinamento è un'informazione che riguarda l'insieme degli elementi e non viene mantenuta all'interno del singolo oggetto. Due esempi di queste tecnologie sono JAXB, XmlSerializer di .NET Framework e Castor.

	Push	Pull	Tree	Cursor	Object to XML Mapping
Forward-Only Access (streaming)	X	X			
Accesso casuale (in memoria)			X	X	X
Richiede un XML-Schema					X
Accesso in sola lettura	X	X			
Basato sugli eventi	X				
Sfrutta il modelli di dati XML	X	X	X	X	

Tabella 4: riepilogo dei modelli per la gestione dei flussi

Parser XML utilizzato

I motivi che ci hanno guidato nella scelta del modello e quindi del parser da adottare sono stati quelli di conformità agli standard, indipendenza dal sistema operativo e interoperabilità con gli altri strumenti del framework. Inoltre è stato deciso di non appoggiarsi alla libreria standard di Java per la gestione dei documenti XML, ma di utilizzare una libreria esterna. La ragione di questa scelta risiede nel fatto che le API per l'XML sono state inserite nel Runtime Environment di Java solo a partire dalla versione 1.4, ed esistono tuttora molti sistemi dove viene utilizzata la versione 1.3 in cui il nostro framework non potrebbe essere utilizzato. Dopo alcune ricerche (limitate al software con licenza non commerciale) sono stati individuati diversi candidati al nostro scopo:

1. GNU Jaxp[42]: una libreria sviluppata nell'ambito del progetto GNU che permette di gestire documenti XML sia attraverso il modello DOM (Document Object Model), che gestisce l'intero documento mantenendone l'intera struttura in memoria, sia attraverso l'interfaccia SAX (Simple API for XML), che permette di definire direttamente dei parser su misura. Inoltre permette di validare i documenti rispetto alle DTD.
2. Lark[43]: la prima libreria sviluppata per gestire i documenti XML,

42 http://www.gnu.org/software/classpathx/jaxp/
43 http://www.textuality.com/Lark/

scritta da Tim Bray (uno dei creatori di XML). Dato che è stata scritta quando ancora non c'era uno standard, le funzionalità fornite da Lark non sono omogenee con quelle delle altre librerie.

3. Piccolo[44]: un parser XML molto compatto e veloce che però implementa solo l'interfaccia SAX e non permette di validare i documenti.
4. Xerces[45]: il parser XML del progetto Apache. Supporta sia DOM che SAX e permette di validare i documenti sia rispetto a una DTD che rispetto a un XML-Schema.
5. XP[46]: una libreria che oltre a SAX permette di definire altri tipi di parser attraverso funzioni di basso livello. Non supporta DOM e non permette di validare i documenti.

Dopo varie considerazioni la scelta è caduta su Apache Xerces. Le motivazioni dietro a questa scelta sono diverse: Lark, XP e Piccolo sono stati scartati immediatamente a causa delle lacune nelle caratteristiche fornite, nonostante Piccolo vantasse ottime performance. Quindi rimaneva da effettuare una scelta limitata tra GNU JAXP e Apache Xerces; alla fine si è deciso in favore di quest'ultimo a causa di una questione di compatibilità: le nuove API introdotte nella libreria standard di Java 1.4, infatti, utilizzano proprio Xerces come implementazione, e si sono mantenute conformi alla segnatura stabilita per quest'ultimo. Questa caratteristica permette quindi di utilizzare il framework sia in ambienti che utilizzano la Java Virtual Machine 1.3 (inserendo la libreria esterna di Xerces), sia in ambienti che utilizzano la Java Virtual Machine 1.4 (affidandosi alle librerie interne), senza dover ricompilare l'applicazione. Dato che Xerces offre il supporto per due API: quella di SAX e quella di DOM, abbiamo potuto utilizzare entrambi gli approcci. Il primo l'abbiamo usato per la visualizzazione dei documenti, fase in cui non è richiesto di potere modificare il flusso

44 http://piccolo.sourceforge.net/
45 http://xml.apache.org/xerces2-j/index.html
46 http://www.jclark.com/xml/xp/

XML, e dove sfruttando l'approccio basati sugli eventi siamo in grado di visualizzare i documenti velocemente (perché i nodi vengono analizzati una volta sola) ed efficientemente (il documento non viene caricato in memoria). Il secondo, l'abbiamo invece impiegato nella fase di editing perché, caricando il documento in memoria, permette di manipolarne i nodi in qualsiasi momento. Per maggiori informazioni su XML si può consultare [Harold2001] oppure le specifiche del World Wide Web Consortium pubblicate sul sito ufficiale[47].

Database XML Nativi

Introduzione

La crescente diffusione del formato XML, sia all'interno delle aziende che nella pubblica amministrazione, pur avendo semplificato le operazioni di condivisione dei dati ed evitato di doversi affidare ad un formato proprietario, ha fatto sorgere la necessità di rendere disponibili degli strumenti che fossero in grado di gestire e processare questo tipo di documenti in modo più efficiente. La prima strada che si è tentata, è stata quella di estendere i database relazionali perché fossero in grado di supportare questo tipo di dato con i cosiddetti database "xml embedded", la seconda è stata quella di realizzare dei database che supportassero nativamente il formato XML, con quelli che poi sono stati chiamati database "XML nativi"[48]. Decidere quale delle due strategie sia la più efficiente o quale sia superiore all'altra non è banale in quanto tale scelta è, come vedremo, fortemente legata dal tipo dei dati che si dovranno rappresentare e dalle funzionalità richieste.

47 http://www.w3.org/XML/

48 Una buona definizione di database XML nativo è "un sistema che definisce un modello logico per i documenti XML e permette di accedervi e memorizzarli in base a questo modello, assume il documento come unità minima di memorizzazione e non vincola a nessun modello fisico di memorizzazione"[Bourret03]

Dati o documenti ?

Il fattore più critico nella scelta del tipo di database da adottare è capire se lo si utilizzerà per memorizzare dati o documenti. Se il formato XML è impiegato esclusivamente come mezzo per trasportare i dati tra il database e un'applicazione, che possibilmente non supporta questo formato, in quel caso parleremo di documento incentrato sui dati (“data-centric”) mentre se viene impiegato nel suo uso integrale come nel caso di XHTML o dei documenti in formato DocBook allora parleremo di documenti incentrati sul contenuto (“document-centric”)[49].

In quest'ultima categoria rientrano tutti i documenti che sono stati progettati per essere compresi dall’uomo come i libri, le e-mail, gli annunci e buona parte dei documenti XHTML scritti a mano. Essi sono caratterizzati da una struttura poco regolare se non completamente irregolare, hanno un alto livello di granularità del dato (ovvero la più piccola forma di dato indipendente può essere un elemento con contenuto misto o l’intero documento) ed hanno parecchio contenuto misto. Vi è poi da dire che questo tipo di documenti normalmente non proviene dai database ma è scritto in XML o in un altro formato come RTF, TXT o PDF e successivamente è convertito in XML.

Viceversa, i documenti incentrati sui dati, sono quelli progettati per essere direttamente gestiti da un'applicazione specializzata, e sono caratterizzati da una struttura abbastanza regolare dove i dati sono molto granulari (ovvero la più piccola unità indipendente è al livello di un elemento che contiene soltanto un PCDATA o di un attributo) e i nodi con contenuto misto non esistono o sono pochissimi. Ciò vuol dire che per l’applicazione o il database non è importante che i dati siano memorizzati in un formato piuttosto che in un altro.

Esempi classici di documenti incentrati sul contenuto sono i libri o gli articoli, la documentazione amministrativa o le pagine web. In tutti

49 Si veda, a proposito, [Bourret03], [Trinidad00] oppure [Kellokoski00]

questi casi si deve gestire un'informazione senza una struttura fissata a priori. Viceversa, esempi classici di documenti incentrati sui dati sono gli ordini di vendita, le prenotazioni di un volo, i dati scientifici o il valore delle azioni quotate in borsa; in generale, nei casi in cui i dati sono omogenei tra loro e sono strutturati secondo uno schema relativamente rigido, questi possono essere gestiti attraverso documenti data-centric.

Nella pratica reale, ovviamente, la distinzione tra queste due tipologie di documenti non è sempre così netta e capita di sovente che un documento incentrato sui dati, come ad esempio la fattura commerciale, possa contenere dei campi dal contenuto altamente irregolare (nel caso della fattura questo potrebbe verificarsi con la descrizione del pezzo fatta attraverso un linguaggio di markup). Viceversa, un documento incentrato sui contenuti come può essere un manuale per l'utente, può contenere dati con una struttura regolare e fissata, come ad esempio il nome dell'autore o la data di revisione.

Quale Database?

A fronte di tutto questo, la scelta del tipo di database da utilizzare non è stata semplice: la norma generale prevede che i dati siano gestiti con i database tradizionali, come quelli relazionali o quelli ad oggetti, e che i documenti siano memorizzati da un database XML nativo o da un sistema di content management, che si trova al di sopra di un database XML nativo. Questa regola non è comunque assoluta, soprattutto perché oggi la linea di demarcazione tra database tradizionali e quelli XML nativi sta cominciando a sparire, dato che nel primo caso si sta cercando di aggiungere delle funzionalità per supportare nativamente XML mentre nel secondo si lavora alla gestione dei frammenti di documenti che provengono dai database esterni (normalmente quelli relazionali). Sebbene il dominio del problema affrontato coinvolga documenti con una

struttura abbastanza regolare, come quella che caratterizza la modulistica, abbiamo scelto di utilizzare come repository per i documenti un database XML nativo piuttosto che un tradizionale database relazionale. Questa scelta è stata dettata dalla necessità di dover accedere alla struttura gerarchica dei documenti considerati in quanto tali, piuttosto che come semplici contenitori di dati.

Database XML nativi

Uno degli standard più diffusi tra questa categoria di database è quello definito dall'iniziativa XML:DB, secondo cui un "Native XML Database" (NXD) deve offrire almeno le seguenti funzionalità [XMLDBFAQ]:

- definizione di un modello logico per i documenti XML;
- le operazioni di lettura e scrittura sui documenti devono avvenire solo in base a quel modello;
- il modello deve almeno includere: gli elementi, gli attributi, i PCDATA e l'ordine dei documenti;
- il documento XML deve essere l'unità minima di accesso, così come la riga di una tabella lo è per i database relazionali;
- non è richiesto che sia presente un particolare modello di memorizzazione fisica. Ad esempio un database XML può essere costruito al di sopra di un database relazionale, gerarchico o ad oggetti, oppure può utilizzare formati di memorizzazione proprietaria come le strutture indicizzate o i file compressi.

In particolare, un database XML:DB si basa sulla concezione delle "collection", collezioni di elementi atomici che svolgono un ruolo molto simile a quello delle tabelle in un modello relazionale. A differenza di queste ultime, però, i documenti contenuti nelle collection non sono vincolati ad essere conformi ad un formato particolare, ed anzi, non è neppure richiesto che due documenti inseriti nella stessa collection siano dello stesso tipo. Questo significa che all'interno di una collection può essere inserito qualsiasi tipo di documento, indipendentemente dal XML-Schema che lo definisce, ed

è comunque possibile eseguire delle query su tutti i documenti (eventualmente disomogenei) della collection; questa caratteristica dei NXD è chiamata “Schema-independence”[Staken03]. Questo rende il database estremamente flessibile, semplificando nel contempo lo sviluppo delle applicazioni, ma il rovescio della medaglia è che questa funzionalità crea non pochi problemi agli amministratori dei database, per l'impossibilità di sapere se i dati all'interno del repository siano corretti, a meno di non utilizzare un tool di verifica (che comunque dovrebbe essere sviluppato appositamente). Queste collection, comunque, sono organizzate logicamente all'interno del database in una struttura ad albero, sulle cui foglie sono posizionati i documenti, mentre gli altri nodi sono occupati dalle collection stesse, in una maniera molto simile all'organizzazione classica di file e directory in un filesystem. Per maggiori informazioni sui database XML:DB, si può consultare la pagina principale del progetto[50] oppure [Bourret03].

Xindice ed eXist

Tra le varie iniziative open source che mirano alla realizzazione di database XML nativi, ve ne sono due che si sono dimostrate abbastanza mature e stabili da essere impiegate in un progetto reale: Xindice, sviluppato da Apache Foundation, ed eXist, scritto da Wolfgang M. Meier. Entrambi sono completamente scritti in Java e ufficialmente mirano alla completa implementazione delle API di XML:DB ma le loro similitudini finiscono qui, visto che ognuno ha utilizzato un approccio leggermente diverso nella gestione dei documenti XML.

Xindice:

Xindice è un database XML nativo, sviluppato dalla Apache Foundation, che implementa lo standard XML:DB. Xindice può essere impostato per essere utilizzato in due modi differenti, ognuno dei

50 http://www.xmldb.org/

quali implica una diversa modalità di accesso. Nella prima, la cosiddetta “modalità embedded”, l'applicazione Java sviluppata può impostare un'istanza di Xindice all'interno della stessa JavaVirtual Machine su cui sta girando, e in questo caso solo quell’applicazione sarà in grado di accedere e di manipolare i dati che si trovano su Xindice, utilizzando direttamente le API XML:DB per accedere al database istanziato. Eventuali altri programmi che volessero accedere al database, invece, dovrebbero forzatamente utilizzare l'applicazione come interfaccia verso il repository dei dati, non avendo la possibilità di accedere direttamente al database. Al contrario, utilizzando Xindice in modalità server, una qualsiasi applicazione può connettersi al database attraverso chiamate di procedura remota che utilizzano lo standard XML-RPC, un protocollo costruito su uno strato superiore ad HTTP. Per quello che riguarda le funzionalità aggiuntive rispetto allo standard XML:DB, Xindice dà la possibilità di indicizzare le singole collection su elementi o attributi dei documenti, in modo da velocizzare le operazioni di ricerca ed infine prevede la compressione dei dati per risparmiare spazio sul filesystem[51]. Nonostante queste ottimizzazioni, comunque, il database non riesce ancora a gestire bene documenti di grosse dimensioni, ma riesce a dare le migliori prestazioni nel caso in cui si debba gestire un grande numero di piccoli documenti. Una delle pecche forse più gravi di questo prodotto, però, è l'impossibilità di gestire i caratteri che non siano ASCII standard[52], e questo impedisce di fatto l'utilizzo delle lettere accentate.

51 Dato che i documenti XML contengono esclusivamente testo, un buon algoritmo di compressione può ridurre lo spazio occupato di una frazione molto significativa

52 Questo, almeno fino alla versione 1.0 del prodotto. È stato annunciato che la versione 1.1 dovrebbe risolvere questo problema

Architettura interna di Xindice[53]

Struttura del database

Logicamente, i documenti XML memorizzati in Xindice sono organizzati in una gerarchia di “collection”. Una “collection”, come il nome stesso suggerisce, è un insieme che può contenere un numero non specificato di documenti e collection. Il fatto che una collection possa contenere altre collection permette di organizzare i dati in maniera gerarchica. La radice di questa struttura, inoltre, è vincolata dal fatto che non può contenere documenti, ma solo collection. Ogni collection del database ha associata una particolare configurazione, che ne specifica l'organizzazione interna. Queste informazioni di configurazione sono contenute all'interno di una collection particolare (“System”), la cui esistenza è sempre garantita. Per ovvie ragioni le informazioni di configurazione di System non sono contenute all'interno di questa collection, ma sono state inserite direttamente dentro al codice, dato che senza di esse non sarebbe stato possibile accedere a System per recuperarle. Fisicamente, la struttura logica della gerarchia di collection è realizzata attraverso il filesystem: ad ogni collection corrisponde una directory, le cui sottodirectory sono esattamente quelle corrispondenti alle collection figlie. I documenti della singola collection, invece, sono solitamente contenuti all'interno di un unico file posizionato all'interno della directory stessa, anche se questo dipende dal particolare Filer selezionato per quella collection (vedi sotto).

Modalità di memorizzazione dei documenti

I documenti non sono memorizzati secondo una struttura predefinita, ma la loro memorizzazione è gestita da una classe (che implementa una particolare interfaccia chiamata “Filer”) definita all'interno delle informazioni di configurazione della collection. In questo modo si possono specificare strutture differenti per collection differenti, in

53 La descrizione della struttura di Xindice si basa sulle specifiche tecniche fornite in [XindiceTech]

risposta ad esigenze di efficienza diverse. Attualmente Xindice fornisce diversi Filer: FSFiler, memorizza i documenti direttamente sul filesystem senza alcuna organizzazione particolare, MemFiler memorizza i documenti in memoria indicizzandoli attraverso una tabella hash e BTreeFiler memorizza i documenti all'interno di un singolo file secondo una struttura B+-Tree. Xindice fornisce anche un HashFiler, che memorizza i documenti in un unico file indicizzandoli attraverso una tabella hash, ma questo Filer è stato deprecato in favore di BtreeFiler, che fornisce prestazioni migliori. BtreeFiler è il Filer utilizzato di default da Xindice, e la sua struttura verrà descritta più in dettaglio.

BtreeFiler memorizza i documenti in un unico file, suddiviso logicamente in pagine di ampiezza fissa per ottenere un accesso più efficiente ai dati (la dimensione della pagina, 4Kb, è inserita hard-coded nel codice della classe). All'inizio del file è presente una pagina di intestazione, che contiene le informazioni necessarie per la gestione della paginazione, come la prima pagina disponibile, il numero dei dati contenuti o la pagina che contiene la radice del B+-Tree di indicizzazione. All'interno del file, i dati sono memorizzati in insiemi di una o più pagine (non necessariamente consecutive), denominati “record”. Nel caso il dato da memorizzare sia più grande di una pagina, viene allocato un numero abbastanza grande di pagine per contenerlo ed ad ognuna di esse viene fornito un puntatore alla successiva., formando una lista di pagine (vedi esempio in figura 2)

Al di sopra di questa struttura a pagine, i documenti sono organizzati secondo un B+-Tree, in cui i puntatori ai nodi e ai dati indicizzati dalle chiavi sono i numeri della pagina in cui inizia il record che contiene il nodo o il dato cercato. In questo modo si ottiene un accesso efficiente ai dati attraverso le chiavi di ricerca. La memorizzazione dei documenti veri e propri, invece, viene fatta trasformando il documento in un “DOM compresso” e memorizzandolo all'interno di un record. Il “DOM compresso”, non è

altro che la serializzazione di un documento XML data dalla sequenza dei nodi e degli attributi del documento (memorizzati nell'ordine in cui vengono letti) in cui i nomi degli elementi sono sostituiti da riferimenti ad una tabella dei simboli, specifica per ogni collection, contenuta all'interno della collection di sistema “System”.

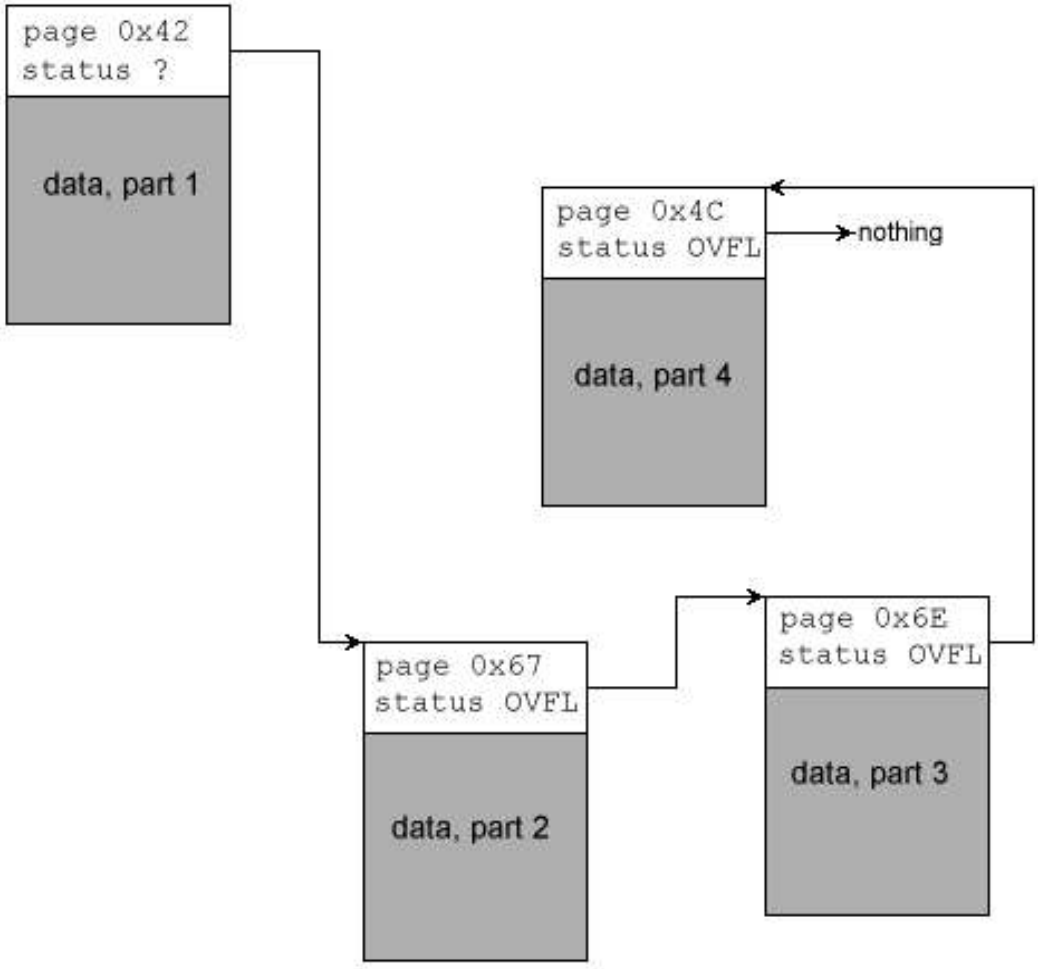

Figura 2: la struttura a pagine di Xindice
(estratto dalla documentazione tecnica)

Struttura delle queries

Il tipo di query possibili su una collection, come nel caso dell'organizzazione dei documenti, non è stabilita a priori, ma è

personalizzabile attraverso la modifica della configurazione della collection. Xindice fornisce di default due motori di interrogazione del database, che implementano lo standard XUpdate e XPath. Dato l'uso esclusivo che ne viene fatto all'interno del framework sviluppato, il motore XPath verrà analizzato in dettaglio.
Xindice non fornisce un'implementazione interna di un motore di valutazione XPath, ma si appoggia alla libreria Xalan di Apache Foundation. Questo significa che per valutare una query XPath, i documenti interessati devono essere caricati in memoria uno alla volta e su ognuno di essi la query è valutata attraverso le classi fornite da Xalan. Infine i risultati vengono aggregati assieme e restituiti. Da quello che si è appena detto, deriva che non c'è alcun guadagno di efficienza, rispetto all'utilizzo del filesystem, nell'utilizzare Xindice per la valutazione di un XPath su un singolo documento. Il contributo principale di Xindice alla velocizzazione delle query XPath, infatti consiste nel restringere lo spettro dei documenti basandosi sulla struttura della query e sull'indicizzazione del database per selezionare un sottoinsieme abbastanza piccolo dei documenti della collection, i quali vengono poi passati al motore di Xalan per la verifica della query.
Per altri approfondimenti si consiglia la pagina principale del progetto[54].

eXist

eXist è un database XML nativo sviluppato da Wolfgang Meier, che implementa in modo quasi completo lo standard XML:DB e supporta la ricerca di testo libero attraverso un sistema di indicizzazione. Quest'ultima caratteristica, pur permettendo delle ricerche estremamente efficienti, può essere utilizzata solo attraverso delle funzioni non standard all'interno delle query XPath, costringendo lo sviluppatore a scegliere tra l'aderenza allo standard XML:DB e la velocità di esecuzione. Per quello che riguarda il resto, anche eXist,

54 http://xml.apache.org/xindice/

come Xindice, può essere utilizzato in modalità “embedded” all'interno dell'applicazione oppure come server a cui si accede in remoto. Per quel che riguarda la sicurezza, bisogna osservare che eXist permette di utilizzare un sistema di permessi sulle singole collection equivalente a quello standard di Unix, con i permessi di lettura, scrittura e aggiornamento[55] dei documenti contenuti nella collection.

Architettura interna di eXist[56]

Al contrario di altri linguaggi di query XML disponibili come software open source, eXist non si basa su algoritmi di visita dell'albero degli elementi per valutare le espressioni di ricerca. La ragione dietro a questa scelta è che questi algoritmi diventano estremamente inefficienti per insiemi che contengono un numero molto grande di documenti. La soluzione generalmente adottata per aumentare l'efficienza è utilizzare strutture indicizzate. Ma mentre le ricerche basate sui contenuti sono solitamente ben supportate estendendo gli schemi di indicizzazione tradizionali (come i B+-tree), le ricerche basate sulla struttura del documento sono più difficili da ottimizzare. Per gestire bene quest'ultimo tipo di ricerca la struttura di indicizzazione utilizzata dovrebbe poter calcolare velocemente relazioni come padre/figlio o antenato/discendente tra nodi differenti. Uno dei metodi proposti in letteratura consiste nel numerare i nodi del documento secondo un certo schema, in modo che le relazioni strutturali possano essere ridotte ad espressioni matematiche parametriche rispetto ai valori dei numeri associati ai nodi. Uno di questi schemi di numerazione consiste nel rappresentare il documento come un albero n-ario completo (dove n è uguale al massimo numero di nodi figli di un qualsiasi nodo del

55 Il permesso di “aggiornamento” sembra più dovuto alla volontà di mantenersi isomorfo ai permessi del filesystem Unix che non a una reale necessità, in quanto equivale ad un permesso di scrittura con il vincolo che il documento esista.

56 La descrizione della struttura di eXist si basa sulle specifiche tecniche fornite in [Meier02]

documento) e assegnare ad ogni nodo un identificatore unico attraversando l'albero breath-first. In figura 3 sono mostrati gli identificatori assegnati ai nodi di un albero binario molto semplice.

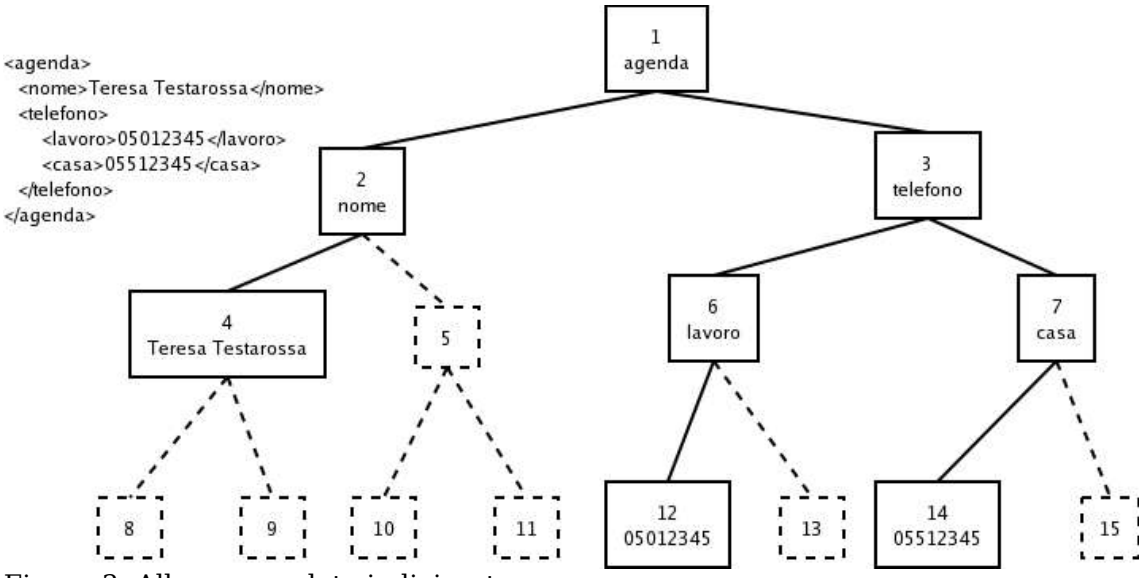

Figura 3: Albero completo indicizzato

I nodi e gli archi tratteggiati indicano elementi che non sono presenti all'interno del documento, ma che devono essere comunque contati, data la necessità di considerare un albero completo. Numerando i nodi in questa maniera, le relazioni strutturali vengono ridotte a semplici operazioni matematiche, per cui, in un albero n-ario completo, identificare l'indice del padre del nodo di indice i si riduce a calcolare il suo indice, attraverso la formula

$$\text{parent}_i = \text{floor}\left(\frac{(i-2)}{n}+1\right) \quad .$$

Il vincolo di completezza, però, porta ad una pesante restrizione alla dimensione massima dei documenti che possono essere indicizzati con questo metodo: ad esempio un documento strutturato come un tipico articolo scientifico avrà un numero limitato di elementi di alto livello (come capitoli o sezioni), mentre la maggioranza dei nodi consisterà in paragrafi e nodi di testo collocati a un livello più basso. In questi casi viene introdotto un gran numero di identificatori

“fantasma”, e questi numeri crescono molto velocemente anche per piccoli documenti. Il metodo di numerazione adottato da eXist fornisce una variazione di questo schema che abbandona parzialmente il vincolo di completezza per evitare una crescita esplosiva della numerazione degli indici. In questo schema di numerazione, il documento non è più visto come un albero n-ario completo, ma il numero di figli di un nodo è il massimo tra le cardinalità dei figli limitatamente al livello di profondità del nodo. L'informazione su quanti nodi figli debbano essere considerati ad un dato livello è contenuta in un array memorizzato assieme al documento. In figura 4 si può vedere lo stesso documento dell'esempio precedente, numerato secondo questo schema.

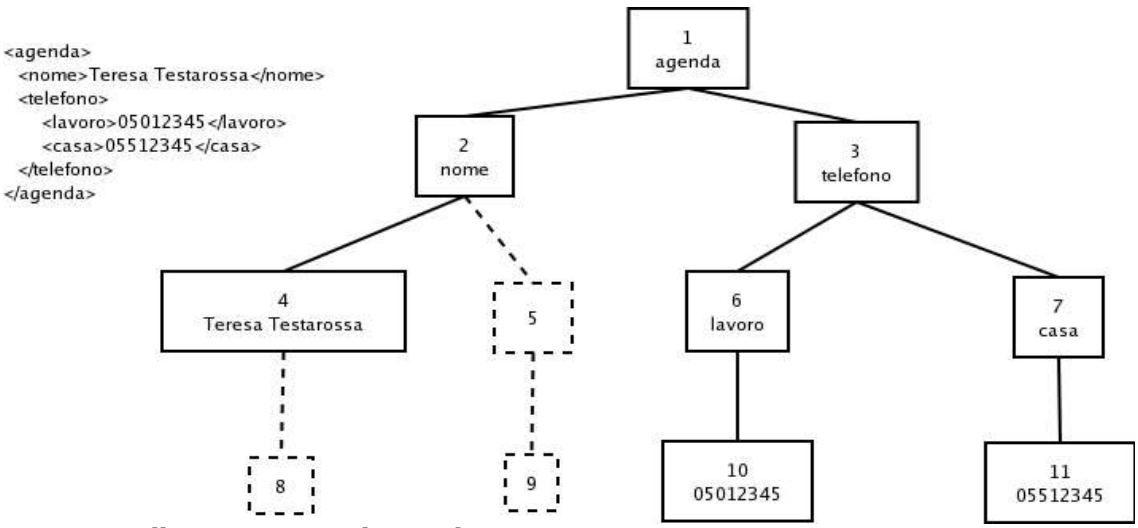

Figura 4: albero non completo indicizzato

L'utilizzo di un sistema di numerazione dei nodi consente una facile navigazione in ogni direzione dell'albero (sia in profondità che in ampiezza) senza dover mantenere puntatori di collegamento tra i nodi, che avrebbero aumentato la dimensione del documento in maniera significativa.

Organizzazione sul filesystem:

eXist mantiene quattro file per la memorizzazione delle informazioni, organizzati internamente come B+-Tree per garantire un accesso

efficiente ai dati contenuti. Un punto importante da sottolineare è che all'interno di questi file, gli indici per gli elementi, gli attributi e per il contenuto sono organizzati per collection, non per documento. Ad esempio tutte le occorrenze di un elemento “section” all'interno di una collection sono riferite dallo stesso indice all'interno di elements.dbx. Questo contribuisce a mantenere la dimensione del B-+Tree entro una dimensione accettabile e velocizza le ricerche che spaziano sull'intera collection (un caso estremamente comune).

Di seguito vengono esaminati questi file più in dettaglio:

- collections.dbx gestisce la gerarchia delle collection e mappa i nomi delle collection sugli oggetti contenuti. Ad ogni collection e ad ogni documento è assegnato un identificatore unico.
- Dom.dbx rappresenta il componente centrale della struttura di memorizzazione di eXist, ed è costituito da un unico file in cui sono memorizzati tutti i nodi dei documenti secondo lo standard DOM del W3C. Questi nodi sono indicizzati da un B+-Tree a radici multiple contenuto nello stesso file che associa, per un dato documento, un identificatore unico di nodo al contenuto del nodo stesso (si veda l'illustrazione in figura 5)

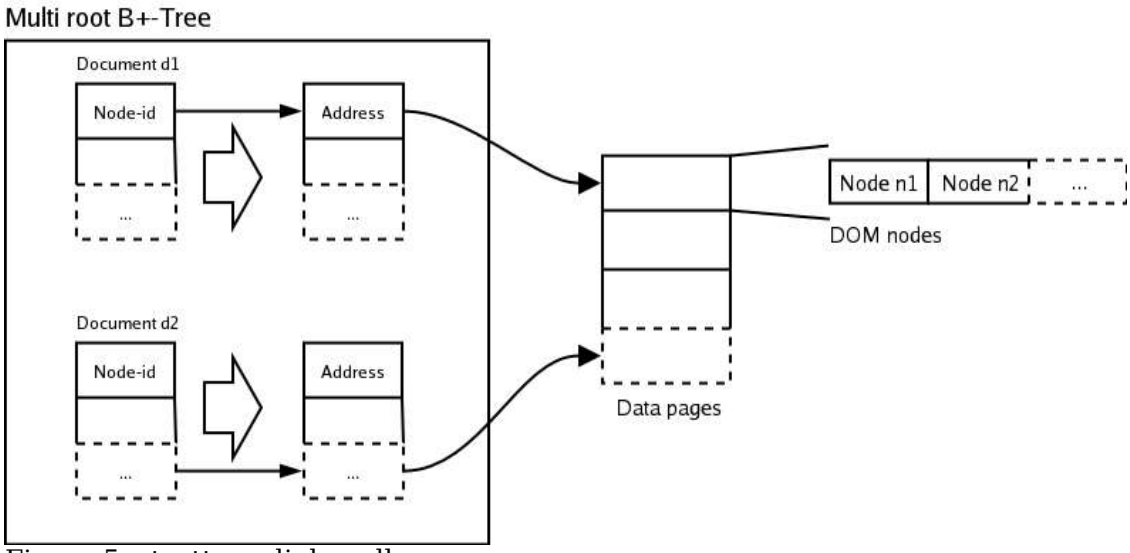

Figura 5: struttura di dom.dbx

Non tutti i nodi vengono indicizzati dal B+-Tree: attributi, nodi di testo e i nodi ai livelli più bassi della gerarchia del documento non sono indicizzati, e l'accesso a questi nodi è consentito dallo schema di numerazione dei nodi visto sopra, a partire dall'antenato più vicino indicizzato dal B+-Tree. Inoltre i nodi sono memorizzati nell'ordine in cui appaiono nel documento in modo che la serializzazione di un documento o di un frammento di documento necessita di un singolo accesso all'indice per trovare la radice del documento o del frammento.

- Elements.dbx mappa i nomi degli elementi e dei nodi agli identificatori unici associati. Ogni elemento dell'indice consiste in una chiave formata da una coppia <collection,nome>, associata ad una lista di identificatori di documenti e di nodi che corrispondono ai nodi con quel nome ai documenti che li contengono (vedi figura 6).

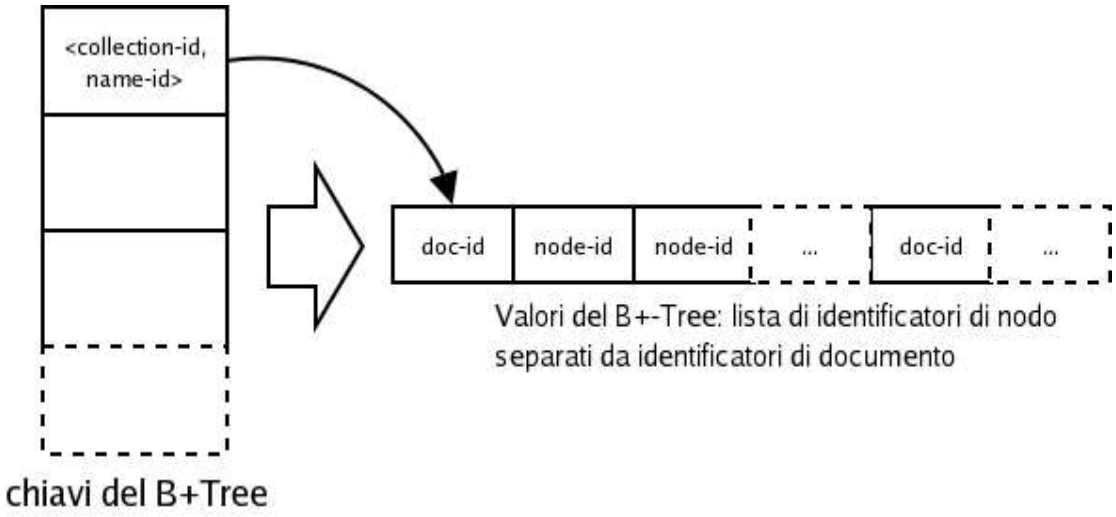

Figura 6: struttura di elements.dbx

Ad esempio, se il motore di ricerca deve recuperare tutti i nodi che puntano ad un elemento “chapter” di un docbook qualsiasi, fra quelli contenuti nella collection specificata, può farlo attraverso un unico accesso all'indice.

- Words.dbx è un indice invertito utilizzato per associare parole o

frasi ad un insieme di documenti e alla posizione in cui il testo si trova all'interno di ogni documento. Questa posizione è indicata dall'identificatore del nodo.

Per maggiori informazioni si può visitare il sito del prodotto[57] oppure consultare [Meier02].

Conclusioni

Durante la realizzazione del nostro progetto sono stati utilizzati entrambi i database, anche per verificare l'effettiva modularità dell'architettura, e nessuno dei due prodotti si è rivelato completamente esente da problemi, nonostante siano entrambi abbastanza stabili e completi da poter essere utilizzati in fase di produzione. Come già accennato, la mancata gestione delle lettere accentate in Xindice pone un grosso ostacolo all'utilizzo da parte degli utenti del servizio che sono costretti ad utilizzare l'apostrofo in sostituzione, ed inoltre si sono riscontrati alcuni problemi con determinate versioni della Java Virtual Machine che rendevano instabile il sistema. L'utilizzo di eXist, invece, non ha portato nessuno di questi problemi, ma si è rilevata una minore efficenza durante la fase di inserimento e aggiornamento dei documenti, dovuta probabilmente all'overhead del sistema di indicizzazione per la ricerca su testo libero e del calcolo degli indici dei nodi nella rappresentazione interna. Inoltre l'implementazione incompleta di XPath costringe ad utilizzare delle funzioni non standard per ottenere risultati equivalenti.

Apache Tomcat

Tomcat è un'implementazione di un web application server open source che implementa le specifiche Java servlet e JavaServer Page, ed è stato utilizzato come base per il nostro framework. La ragione di questa scelta consiste nel fatto che Tomcat è un Java application server open source, ben documentato, aderente agli standard e con

57 http://exist.sourceforge.net/index.html

una base di utenza molto grande che avrebbe potuto essere d'aiuto qualora si fossero riscontrati dei problemi durante lo sviluppo del framework. Inoltre Sun utilizza proprio Tomcat per l'implementazione di riferimento delle servlet e delle Java Server Pages (JSP) all'interno di Java 2 Enterprise Edition, facendone in pratica l'applicazione standard a cui gli altri Java application server dovrebbero allinearsi.

Storia[58]

Il progetto Tomcat ha una storia breve ma interessante: quando Sun rilasciò le specifiche di JSP 1.0 e servlet 2.0, decise di sviluppare una sua implementazione di riferimento integrata pesantemente con il proprio web server. Contemporaneamente a questa decisione un gruppo di programmatori open source decise di creare un engine JSP/servlet che fosse compatibile con le specifiche rilasciate da Sun e che si integrasse con Apache, uno dei web server più diffusi al mondo. Al contrario dell'implementazione di Sun, che sacrificava l'efficienza per l'aderenza allo standard, il progetto Apache JServ (come era stato chiamato) tendeva a focalizzare gli sforzi sull'ottenere un prodotto che avesse buone prestazioni in termini di velocità e stabilità. Gli utenti si trovavano quindi a dover scegliere tra un'implementazione lenta ma aderente allo standard e una più veloce ma che non seguiva tutte le specifiche. Questa situazione durò fino al 1999, quando Sun decise di donare il sorgente della sua implementazione di riferimento alla Apache Software Foundation, che formò un sottogruppo denominato "Jakarta", incaricato di fondere il prodotto Sun con JServ. A un certo punto, però, tutti i progetti Java della fondazione Apache finirono per passare sotto il controllo del gruppo Jakarta, che rinominò il suo progetto principale "Tomcat". L'implementazione di riferimento Sun diede origine a Tomcat 3, mentre con Tomcat 4 venne implementata una nuova architettura ad alte prestazioni che supportasse le specifiche 1.1

58 Questo paragrafo è basato su [Li01]

delle JSP e 2.2 delle servlet. Attualmente è in fase di sviluppo la versione 5 di Tomcat, che dovrebbe migliorare scalabilità ed efficienza, oltre a fornire una migliore gestione della sicurezza e alcune funzionalità che ne aumenteranno l'usabilità.

Funzionalità

La funzione principale del server Tomcat è quella di fornire la possibilità di utilizzare programmi Java lato server per applicazioni web. Si occupa quindi di:

- gestire le richieste (tipicamente inviate attraverso il protocollo HTTP) che provengono da un programma client esterno.
- Eseguire le operazioni necessarie per soddisfare la richiesta ricevuta attraverso l'utilizzo di servlet o JSP (vedi sotto).
- Restituire la risposta al client che ha inviato la richiesta.

Oltre a questo, Tomcat fornisce anche una serie di funzionalità accessorie quali:

- gestione automatica del multithreading, utilizzando un pool di thread per garantire buone prestazioni.
- Possibilità di logging degli eventi durante l'esecuzione della web application.
- Possibilità di utilizzare connessioni SSL[59].
- Autenticazione degli utenti[60].

Come si è detto poco sopra, le operazioni necessarie per soddisfare una richiesta ricevuta dal client possono essere eseguite attraverso l'utilizzo di due tecnologie leggermente differenti denominate rispettivamente servlet e JavaServer Pages (JSP), e la scelta tra le due dipende principalmente da quanta informazione deve venire generata dinamicamente per formare la pagina web inviata al client.

---

59 SSL (Secure Socket Layer), è una tecnologia che consente ai web browser di comunicare con i web server su una connessione resa sicura atttraverso meccanismi di crittazione. Da notare che il meccanismo di SSL funziona nelle due direzioni: questo significa che i dati, prima di essere inviati, vengono crittati sia dal server che dal client[Apache02].

60 Per maggiori informazioni, si può visitare: http://jakarta.apache.org/tomcat/index.html

Servlet

La tecnologia Java servlet sviluppata da Sun, fornisce agli sviluppatori di applicazioni web un meccanismo semplice e consistente per estendere le funzionalità di un web server. Una servlet, infatti, può essere praticamente considerata come una applet che gira lato server, con il vantaggio che, al contrario di una applet, una servlet può utilizzare facilmente tutte le API fornite da Java senza limitazioni, visto che, essendo codice creato dagli sviluppatori dell'applicazione web, non presenta i problemi di sicurezza che invece affliggono un codice scaricato da una fonte esterna ed eseguito lato client. Oltre a questo, le servlet hanno a disposizione una libreria di funzioni specifiche per il protocollo HTTP in modo da potersi interfacciare facilmente con i browser web.

Secondo la definizione di Sun, una servlet è “un componente web basato su Java che genera dinamicamente un determinato contenuto e che interagisce con i client web attraverso un paradigma di richiesta/risposta implementato da una specifica estensione del web server”[Coward01]. Come funzionalità, le servlet si collocano a metà strada tra le Common Gateway Interface (CGI) e le estensioni specifiche dei server, come la Netscape Server API o i moduli Apache, fornendo però, rispetto a queste soluzioni, tutta una serie di vantaggi[61]:

- In generale sono molto più veloci degli script CGI, dato che utilizzano un differente modello dei processi: una servlet non gira come processo separato, e questo elimina l'overhead di creare un nuovo processo per ogni richiesta che arriva al server.
- Usano un API standard supportata da molti web server.
- Hanno tutti i vantaggi del linguaggio di programmazione Java, inclusi la facilita di sviluppo, l'indipendenza dalla piattaforma su cui girano e la disponibilità dell'enorme insieme di librerie

61 come viene sottolineato nel paragrafo 1.4 di [Coward01], "Comparing servlet with others technologies"

standard del linguaggio.

Il normale funzionamento di una servlet si descrive in maniera estremamente semplice: il client web invia una richiesta HTTP al server indicando la URI che specifica la servlet invocata. A quel punto il Servlet Engine invoca il metodo doGet o doPost della classe (a seconda che la richiesta sia stata una get o una post), all'interno del quale viene generata dinamicamente la pagina che deve essere visualizzata. Questa pagina viene poi convogliata in uno stream di uscita che il web server invia al client. Le variazioni a questo scenario base sono innumerevoli: dato che la servlet è una classe Java come le altre, senza alcuna limitazione a parte quella di implementare una determinata interfaccia, al suo interno è possibile compiere ogni genere di operazione permessa dal linguaggio. Alcuni esempi di queste operazioni possono essere l'interrogazione di un database, l'interazione con il filesystem, l'invio di posta elettronica tramite un mail server, il parsing di espressioni complesse o la validazione dei dati ricevuti dal client. È addirittura possibile limitare il lavoro della servlet al solo calcolo, demandando la visualizzazione della risposta ad un'altra pagina attraverso un'operazione di reindirizzamento, ed infine non è neanche obbligatorio utilizzare il protocollo HTTP, dato che, anche se di solito la classe HTTPServlet è quella più utilizzata, esiste una classe più generale (da cui HTTPServlet deriva), che può essere estesa per creare servlet che utilizzino altri protocolli di comunicazione. Per maggiori informazioni sulle specifiche delle servlet si consiglia [Coward01], mentre un ottimo tutorial introduttivo si trova nel capitolo 15 di [Armstrong03]

JavaServer Pages

La tecnologia delle JavaServer Pages (JSP) è un'estensione della tecnologia servlet, che consente di inserire codice Java all'interno di pagine web in un modo simile a quello di alcuni linguaggi di scripting come Javascript o PHP, così da poter gestire dinamicamente il

contenuto delle pagine visualizzate. Uno dei punti di forza delle JSP è quello di consentire una separazione netta fra il contenuto statico della pagina e quello generato dinamicamente. Infatti, anche se in teoria è possibile trasformare una JSP in un blocco di codice Java che contiene informazioni di presentazione, in pratica conviene sempre limitare al minimo le sezioni di calcolo all'interno della pagina; in questo modo il responsabile del design della pagina può lavorare tranquillamente sulla parte statica senza il timore di introdurre malfunzionamenti nel codice scritto dagli sviluppatori e l'intera web application risulta più facilmente manutenibile. Per ottenere questa separazione le JSP utilizzano due tecnologie differenti: l'integrazione con i JavaBeans e le Tag Extensions:

- JavaBeans: Lo scopo delle API JavaBeans è quello di definire un modello di componenti software per Java, che abbiano un'interfaccia con l'esterno ben definita e che possano essere utilizzati senza preoccuparsi di come siano implementati. Nati inizialmente per essere dei componenti utilizzabili da tool di programmazione visuale[62], i JavaBeans hanno trovato altri utilizzi in svariate aree, una delle quali è proprio quella delle JSP. Ogni JSP, infatti, può aver associato uno o più bean che processano i parametri passati alla pagina dal client oppure forniscono dati inseriti in precedenza, secondo un meccanismo estremamente semplice da usare: innanzitutto bisogna ricordare che la caratteristica principale dei JavaBeans è quella di avere dei metodi standard per scrivere e leggere i campi interni alla classe che si vogliono rendere visibili all'esterno. Nel caso delle JSP questi campi devono avere esattamente lo stesso nome dei parametri passati alla request, i cui valori potranno essere quindi passati (e processati) al bean attraverso una direttiva particolare. I risultati del calcolo potranno poi essere utilizzati in un altro punto della

62 Le specifiche ufficiali della Sun li definiscono come "un componente software riutilizzabile che può essere manipolato in modo visuale da un tool di sviluppo"[Graham97]

stessa JSP oppure in un'altra pagina, a seconda dello scoping assegnato al bean (che può non essere limitato alla pagina, ma può estendersi a tutta la sessione o addirittura a tutta la vita dell'applicazione). In figura 7 si vede lo schema di un possibile scenario che coinvolge l'uso dei beans: il valore passato nella request dal client è un valore mnemonico che indica un colore e che viene passato al bean esattamente come arriva. All'interno del bean, questa stringa viene trasformata in un valore numerico comprensibile all'HTML ed è quest'ultimo che viene restituito alla JSP quando questa ne fa richiesta.

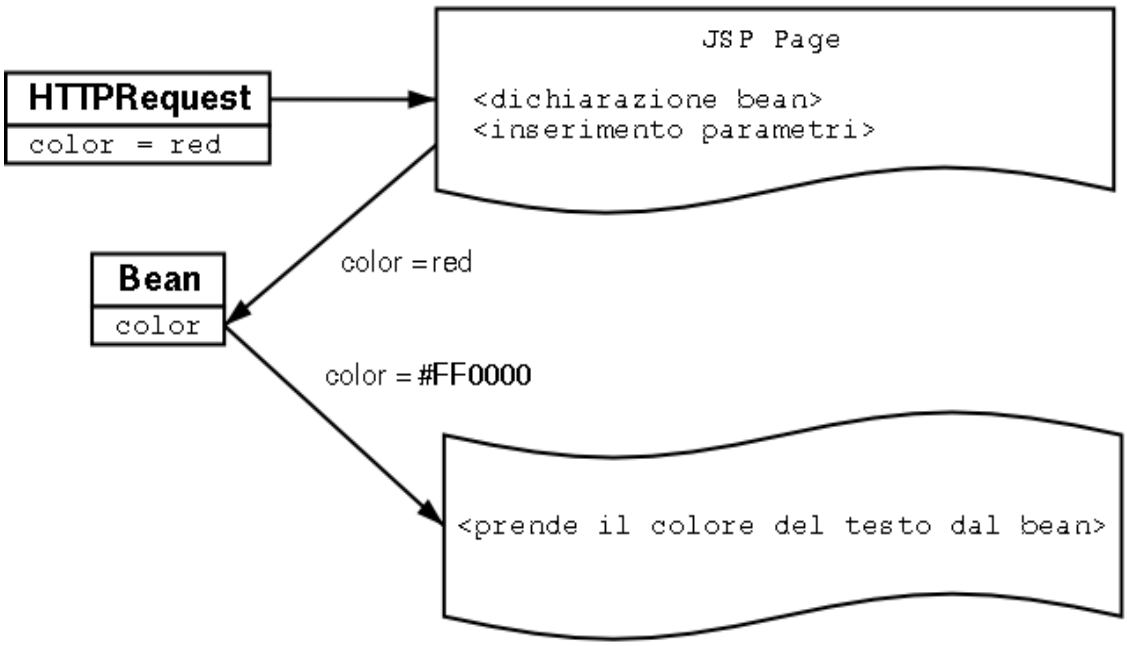

Figura 7: esempio di uso dei beans

- Tag Extensions: Le specifiche JSP, oltre ad integrare l'uso dei JavaBeans, permettono di definire dei tag personalizzati che implementano la logica del contenuto dinamico della pagina. Questi “custom tag” operano un'astrazione delle funzionalità della JSP, definendo un (sotto)linguaggio che consente un utilizzo più naturale di queste funzioni all'interno della pagina stessa [Pelegri01]. La creazione di una Tag Extension è solo leggermente

più complessa di quella di un bean: ogni tag è associato ad una classe Java, la quale riceve in entrata i parametri assegnati al tag e invia su un determinato stream di uscita il codice HTML generato dinamicamente, in maniera simile a quanto avviene per le servlet. Questo codice, poi, va a sostituire il tag nella pagina finale inviata dal server al client web che l'aveva richiesta. L'associazione tag/classe è fatta attraverso un particolare XML di configurazione che viene letto dall'application container.

Per approfondire la conoscenza delle JSP si consiglia [Pelegri01] e il capitolo 16 di [Armstrong03]

Confronto tra servlet e JSP

Come si è detto nel paragrafo precedente, Tomcat mette a disposizione due modi differenti per creare pagine con un contenuto dinamico, ed ognuno di essi ha i suoi punti di forza e le sue debolezze. Innanzitutto è necessario puntualizzare che l'efficienza dei due metodi è pressappoco la stessa, visto che le JSP vengono compilate in servlet da Tomcat prima di essere utilizzate; quindi il confronto va fatto sulle funzionalità offerte e sulla facilità di sviluppo. Le servlet sono certamente la soluzione più flessibile tra le due, dato che consentono un controllo completo sulla pagina che viene generata, ma questo vantaggio può diventare allo stesso tempo un grosso svantaggio per lo sviluppatore, che deve preoccuparsi di scrivere il codice da inviare allo stream di uscita per l'intera pagina, specie se non si implementa una struttura che consenta di separare in modo "pulito" la presentazione dalla logica del programma. Per quello che riguarda le JSP, invece, è possibile specificare solo le parti della pagina da generare dinamicamente, lasciando definito staticamente il resto, e con un uso intelligente dei bean e delle Tag Extensions è possibile separare la presentazione dal calcolo in maniera abbastanza elegante. Nonostante queste considerazioni, le JSP non sono sempre la soluzione migliore. Infatti, dato che per

natura sono meno flessibili delle servlet, quando si ha la necessità di generare comunque l'intera pagina dinamicamente oppure quando si vogliono eseguire operazioni particolari, queste ultime si rivelano di solito la scelta più adatta. In generale, l'esperienza svolta durante lo sviluppo del framework ha evidenziato che per le pagine in cui la presentazione è preponderante rispetto al calcolo, è conveniente utilizzare le JSP, mentre nel caso contrario conviene comunque utilizzare le servlet.

Tomcat come web container

Apache Tomcat è un'applicazione che gestisce delle web application in accordo alle specifiche J2EE[63] di Sun (web container). “Una web application è un insieme di Servlet, JSP, pagine HTML, documenti XML, fogli di stile XSL e CSS, etc. logicamente correlati che offrono un'applicazione o un servizio web completo" [Mamei00]. All'interno di Tomcat, le politiche di sicurezza e le informazioni di configurazione sono gestite separatamente per ogni web application differente. Ogni web application, infatti, per essere utilizzata da Tomcat, deve passare attraverso la cosiddetta fase di “deploy”, cioè devono essere compiute tutte quelle operazioni necessarie per registrarla nel web container, in modo che quest'ultimo possa renderla disponibile ed usufruibile dalle applicazioni client. Innanzitutto l'applicazione deve essere contenuta in un'unica directory, che deve contenere una directory speciale chiamata WEB-INF, la quale non fa parte della gerarchia di documenti pubblica visibile direttamente al client, ma è visibile esclusivamente al codice applicativo. In questa locazione, quindi, lo sviluppatore può inserire informazioni specifiche che non desidera siano visibili al client.

63 “Java 2 Enterprise Edition rappresenta uno standard inteso a facilitare lo sviluppo di applicazioni multi-tier di tipo enterprise. Questo tipo di applicazioni sono tipicamente costituite da un Client-tier che fornisce l'interfaccia utente, uno o più moduli middle-tier che forniscono i servizi veri e propri e la logica applicativa (business logic), e da un backend-tier, costituito tipicamente da un database Server o più in generale da un Enterprise Information System (EIS), che permette la gestione persistente dei dati”.[Mamei00]

WEB-INF deve contenere[64]:

- il descrittore di deployment (web.xml): un documento che contiene varie informazioni necessarie ad interfacciare la web application con il web container, come la segnatura delle servlet (quali parametri accettano e quali sono obbligatori), il nome logico associato ad una determinata servlet, la classe associata a quest'ultima, il mapping dei tipi MIME, la configurazione di sessione, etc.
- La directory classes: una directory che contiene le servlet e le classi ausiliarie di utilità
- La directory lib: una directory che contiene le librerie java utilizzate dal codice applicativo della web application.

Una web application può essere inoltre compressa in un unico file che viene utilizzato da Tomcat in maniera identica alla struttura “espansa” delle directory. L'unica differenza, in questo caso, è che il file compresso deve includere una directory chiamata “META-INF”, contenente delle informazioni necessarie ai tool di compressione Java, secondo le specifiche del formato JAR.

XSL

Storia e introduzione

XSL (eXtensible Stylesheet Language) era stato inizialmente sviluppato dal World Wide Web Consortium come un linguaggio che consentisse la formattazione di documenti XML in maniera indipendente dal mezzo di destinazione. Durante la fase di specifica, però, la parte del linguaggio relativa alle trasformazioni da un tipo di documento all'altro aveva iniziato ad attirare un notevole interesse da parte degli sviluppatori e delle aziende (Microsoft per prima), che ne vedevano l'utilità intrinseca a prescindere dal linguaggio di formattazione vero e proprio e che erano interessate a svilupparne un'implementazione indipendentemente dal linguaggio di

64 Come specificato nel capitolo 9 di [Coward01], “Web Applications”

formattazione dei documenti[65]. Tutta questa attenzione da parte della comunità ha portato alla divisione del progetto originario in tre parti relativamente indipendenti: un linguaggio funzionale che permette di trasformare documenti XML in altri documenti XML, chiamato XSLT (XSL Transform), un linguaggio di query per la ricerca di elementi all'interno di un documento, denominato XPath, e l'insieme originario di elementi di markup che definivano la formattazione del documento in maniera indipendente dal medium di visualizzazione, chiamato XSL-FO (XSL Formatting Objects).

Nonostante questa divisione, i diversi progetti hanno mantenuto delle forti scelte di progettazione in comune [Deach02]:

1. Sintassi: sia XSLT che XSL-FO sono espressi con una sintassi in XML puro, in modo da poter essere manipolati già dall'inizio attraverso gli strumenti esistenti, stabili e testati che sono stati sviluppati per il parsing e la manipolazione di XML e che avevano raggiunto un notevole grado di maturità nel periodo in cui veniva progettato XSL.
2. Dichiarativi: sia XSLT che XSL-FO sono stati progettati come linguaggi dichiarativi. Ciò vuol dire che con un insieme di elementi e di attributi deve potere essere completamente descrivibile l’aspetto del risultato desiderato, al contrario di altri linguaggi che adottano uno stile procedurale[66], che contiene cioè degli algoritmi che indicano come viene prodotto il risultato. È inoltre previsto che si possano usare le espressioni in alcuni attributi, senza dovere soddisfare a requisiti particolari o dovere utilizzare degli script interni.
3. Costruiti su CSS: XSL è stato pensato per estendere e aumentare le funzionalità di CSS-2, in modo da poter essere utilizzato in campi dove quest'ultimo era palesemente inadeguato, ma per rendere meno traumatico il passaggio da uno standard all'altro,

65 Si veda, al proposito, [Gushee03]
66 Come ad esempio Postscript

XSL WorkGroup ha stabilito che il nuovo prodotto avrebbe dovuto condividere il maggior numero possibile di proprietà di stile e di modelli di formattazione con CSS

4. Cross-Media: XSLT deve essere in grado di interagire in maniera stretta con XSL-FO, dato che quest'ultimo deve essere in grado di coprire i requisiti di base per la stampa dei documenti; di supportare un ampio numero di periferiche, includendo quindi funzionalità per il riadattamento o la reimpaginazione per le periferiche wireless e soddisfacendo i requisiti per l'accessibilità che adesso sono richiesti obbligatoriamente da molti governi. Per offrire queste funzionalità, è necessario operare delle trasformazioni attraverso un meccanismo in grado di riorganizzare e filtrare il contenuto sia per offrire una migliore rappresentazione per ogni media, che per soddisfare i requisiti di accesso. La separazione tra le due funzioni (quella di trasformazione e quella di formattazione), inoltre, promuove una separazione dello schema di layout dal controllo sulla sequenza della paginazione, cosicché un contenuto con un determinato stile possa essere inserito in layout differenti.

Maggiori dettagli sulla storia di XSL possono essere trovati su [Froumentin00] oppure, con un livello di dettaglio ancora maggiore, su [Cover03]. Per una panoramica su XSL in generale, un'ottima fonte è il sito del W3C[67].

XPath

XPath è un linguaggio di navigazione progettato dal W3C per consentire a XSLT e ad XPointer di puntare agli elementi che costituiscono un documento XML. XPath è quindi il risultato dello sforzo compiuto dal W3C di offrire una sintassi ed una semantica comune alle funzionalità che sono condivise da XSLT e da XPointer. Il W3C, allo scopo di agevolarne l'impiego ha inoltre deciso di inserirvi delle operazioni di base che permettono di manipolare

67 http://www.w3.org/Style/XSL/

stringhe, numeri e valori di verità. Al contrario di XSLT e XSL-FO, XPath utilizza una notazione compatta, non XML, per facilitarne l'impiego negli URI o nei valori degli attributi, che prende spunto proprio dalla sintassi utilizzata nelle URL o nei filesystem: una delle componenti fondamentali di una query XPath, infatti, è quello che viene chiamato “location path”, una stringa che indica un determinato insieme di nodi attraverso il percorso da seguire sulla struttura del documento XML; un esempio di “location path che può aiutare a spiegare meglio il concetto è la stringa “../bros/*” indica l'insieme di tutti i figli dei fratelli di nome “bros” del nodo corrente[68]. Oltre che per accedere ai nodi interni di un documento XML, questo linguaggio è stato anche progettato per verificare se un nodo corrisponde o meno ad un determinato pattern attraverso un'espressione booleana che può essere anche notevolmente complessa. Queste espressioni possono essere combinate a piacere con i location path, fornendo un linguaggio di navigazione estremamente completo, come si può vedere dalla seguente espressione XPath: “//persona[indirizzo@censito='true' and cittadinanza='italiana']/anagrafica” che seleziona tutti i nodi “anagrafica” figli delle persone che hanno un indirizzo censito[69] e la cittadinanza italiana. XPath modella tutti i documenti XML come se fossero alberi al cui interno ci sono vari tipi di nodi, definendo un modo per calcolare il valore di ogni tipo di nodo, indipendentemente dal fatto che questo sia un elemento, un attributo o del semplice testo. A questo proposito è bene osservare che XPath supporta completamente i Namespace XML e che pertanto il nome di un nodo viene sempre rappresentato da una coppia composta dalla parte locale e dal valore del namespace URI, che eventualmente può essere vuoto. Per maggiori informazioni, si può consultare

68 Con '..' viene indicato il padre nella gerarchia ad albero, mentre * è una wildcard che indica “tutti gli elementi”, uniformemente alla sintassi usata nei filesystem

69 Con il simbolo '@' XPath indica che il nome seguente è quello di un attributo

[ClarkDeRose99]

XSLT

XSLT è un linguaggio funzionale Turing-completo[70] che manipola documenti XML. Il linguaggio stesso è definito in XML ed in effetti i programmi XSLT non sono altro che documenti costruiti secondo un determinato XML-Schema. Il linguaggio trasforma i documenti definendo delle regole di matching, in cui sono inserite le azioni da compiere quando il documento in ingresso contiene determinati tag o attributi; la selezione delle parti su cui avviene il match è fatta attraverso il linguaggio di query XPath, descritto nel paragrafo precedente, che garantisce ad XSLT una notevole flessibilità. Oltre a questo, si possono definire delle funzioni che incapsulino determinate funzionalità in modo da modularizzare il programma, ed in un recente articolo è stato mostrato come sia possibile esprimere la composizione di funzioni, il lambda calcolo, il currying e l'applicazione parziale di funzioni in XSLT[Novatchev02]. Comunque, nonostante la sua potenza espressiva, il linguaggio è estremamente semplice da usare, anche se il fatto di essere esso stesso espresso attraverso documenti XML lo rende alquanto dispersivo e poco compatto, aumentando le difficoltà nell'individuazione degli errori, specie quando si ha a che fare con fogli di stile complessi. Inoltre c'è da dire che per implementare in XSLT alcuni costrutti, come ad esempio i cicli, è necessario avere un minimo di esperienza con i linguaggi funzionali puri e con la ricorsione, per poter sopperire alla mancanza di uno stato e di costrutti imperativi come il while. Nonostante questi inconvenienti, comunque, la capacita di XSLT di trasformare documenti anche molto complessi in maniera potente ed elegante lo rende uno strumento quasi indispensabile per presentare XML sul web o su altri tipi di media. Ulteriori informazioni su XSLT si possono trovare sul sito ufficiale [Clark99], mentre una buona

70 una dimostrazione di completezza puo essere trovata su [Unidex01], dove viene descritta l'implementazione di una macchina di Turing universale in XSLT

introduzione al linguaggio è il capitolo 9 di [Armstrong03].

Xalan

Per gestire le trasformazioni XSLT è stato scelta la libreria Xalan di Apache Foundation, in quanto si tratta di un software stabile, veloce, open source, perfettamente integrato con Tomcat e Xerces e utilizzato come libreria standard per le API di trasformazione XML di Java da Sun stessa. Oltre a tutto questo, Xalan implementa completamente sia le specifiche XSL Transformation 1.0 che XML Path Language (XPath) 1.0 e può operare indifferentemente sia su documenti DOM che attraverso parser SAX. Ma una delle funzionalità forse più utili di Xalan è quella che permette di accettare come fonte di ingresso o di uscita un qualsiasi stream testuale. Questo significa, ad esempio, che il processore XSLT può leggere il documento da un file, da una stringa o da un URL remoto, processarlo attraverso uno o più fogli di stile XSLT[71] e restituirne la rappresentazione testuale su un altro stream di testo qualsiasi, come ad esempio il canale di uscita di una servlet verso il client web. Quest'ultimo caso è particolarmente frequente quando si vuole visualizzare un documento XML su una pagina Web, ed è la soluzione standard per separare la gestione del contenuto dalle informazioni di presentazione quando si utilizzano le servlet. Per maggiori informazioni, si può visitare il sito di Xalan presso la Apache Software Foundation[72].

XSL-FO[73]

XSL Formatting Objects, (abbreviato solitamente in XSL-FO) è il linguaggio di formattazione indipendente dal medium di pubblicazione che il W3C ha progettato per ovviare ai limiti dei Cascading StyleSheet (CSS). Questi ultimi, infatti, erano adatti solo alla formattazione di documenti HTML per il web, mentre l'avvento

71 Xalan da la possibilità di concatenare in maniera efficiente più trasformazioni in una pipeline
72 http://xml.apache.org/xalan-j/index.html
73 Questa descrizione è basata su [Deach02]

di XML come formato standard di rappresentazione dei dati necessitava di qualcosa di estremamente più potente e flessibile. Tra le pecche più gravi di CSS c'erano la completa mancanza di un supporto per la paginazione[74] e l'impossibilità di gestire documenti con strutture mediamente complesse, come i contenuti delle tabelle, le note a margine, gli indici o una suddivisione in articoli del contenuto della pagina, che comunque sono spesso presenti nell'ambito dell'editoria o dei cataloghi per i prodotti reali. Inoltre CSS, pur essendo in grado di descrivere perfettamente la formattazione di un documento web, diventa penosamente povero di caratteristiche quando si prova ad utilizzarlo per un layout destinato, ad esempio, alla stampa. I tag di markup XSL-FO descrivono in modo estremamente dettagliato lo stile di un documento, includendo l'organizzazione del suo contenuto, il layout e le regole di selezione di quest'ultimo (in pratica tutto quello che è necessario per formattare e impaginare un documento). XSL-FO è inoltre in grado di supportare funzionalità come le note, gli indici, le tabelle dei contenuti e l'aspetto delle pagine in base al fatto che siano pari/dispari per la gestione dei documenti dipendenti dai contenuti come i giornali, le riviste o le presentazioni. Una particolare attenzione è stata dedicata alla qualità tipografica e al supporto multilingue: XSL-FO dovrebbe almeno riscontrare se non superare le funzionalità di tipografia e di layout offerte dagli attuali programmi di impaginazione e dovrebbe essere compatibile con il loro modello di formattazione. Il supporto multilingue di XSL-FO, invece, include sia l'uso delle regole di Unicode che di quelle di Unicode bidirezionale senza eccessive difficoltà. Infine XSL-FO prevede di supportare le funzionalità multi lingue adottate nei font moderni come i font OpenType. Al contrario di quello che si potrebbe pensare, la scelta di inserire un livello di dettaglio così ricco all'interno di

74 Per correttezza c'è da dire che la versione 2.0 di CSS ha aggiunto un rudimentale supporto alla paginazione, ma il suo utilizzo non è ancora diffuso, dato che questa caratteristica non è stata ancora implementata da tutti i browser

XSL-FO non ne penalizza affatto l'utilizzo. Infatti i documenti XSL-FO non vengono quasi mai creati manualmente, ma di solito sono il risultato dell'applicazione di un foglio di stile XSLT al documento da formattare. Questo permette di ottenere due vantaggi:

- una volta creato il foglio di stile per un tipo di documento, questo può essere utilizzato a prescindere dal contenuto dei documenti stessi per trasformare automaticamente gli XML in XSL-FO.
- La suddivisione in template del foglio di stile aiuta a focalizzare la formattazione dei singoli elementi, senza dovere ogni volta considerare l'intero documento.

Per maggiori approfondimenti su XSL-FO, si possono consultare le specifiche ufficiali in [Adler01] oppure il capitolo 18 di [Harold2001]

FOP

Tra i vari strumenti oggi disponibili sul mercato per trasformare un documento dal formato XSL-FO in un formato di stampa come PDF o Postscript, la nostra scelta è ricaduta sul progetto FOP lanciato dal gruppo Apache. Oltre che per ragioni economiche[75], abbiamo deciso di utilizzare questo tool perché, pur non offrendo un supporto completo delle specifiche rilasciate dal W3C, è uno dei pochi software di questo genere indipendenti dal sistema operativo, ed inoltre si è rivelato perfettamente integrabile con il resto delle altre tecnologie che abbiamo scelto di adottare. Gli obiettivi di Apache XML FOP Project[76] sono quelli di offrire un formattatore capace di trasformare un XSL FO in un PDF che sia almeno conforme al livello Basic descritto nella W3C Candidate Recommendation del 21 Novembre 2000 e che supporti le specifiche del formato Portable Document Format Specification (Version 1.3) rilasciate da Adobe Systems l’undici Marzo del 1999.

FOP, giunto alla release 0.25 lo scorso Giugno, può essere utilizzato

75 Come tutti i prodotti di Apache Foundation anche FOP è rilasciato sotto una licenza open source

76 http://xml.apache.org/fop/index.html

sia da riga di comando sia invocando direttamente delle specifiche API Java dall'interno del codice. Quest'ultima caratteristica riveste una notevole importanza nello sviluppo di applicazioni web multipiattaforma che hanno bisogno di un modulo per la stampa su carta dei documenti indipendente dal sistema ospite. Per implementare questa funzionalità, infatti, basta convertire il documento XSL-FO, ottenuto applicando un adeguato foglio di stile al documento da stampare, in un formato standard come PDF o Postscript, che possono essere stampati su praticamente ogni piattaforma. È da osservare che a partire dalla versione 0.25 è stato introdotto un meccanismo di base per la crittazione dei documenti renderizzati in PDF, funzionalità che impedisce di vedere, stampare, editare o copiare il testi dal documento.

Capitolo 5: Soluzione proposta

In questo capitolo verranno presentate le ragioni che ci hanno guidato nella scelta della soluzione proposta e le varie parti che la compongono. Innanzitutto verrà definito cosa siano un sistema di Content e Knowledge Management, poi verrà descritta l'architettura del framework e i moduli che lo compongono. Successivamente saranno illustrate le funzionalità offerte dal framework e infine verrà mostrato come il framework soddisfi i requisiti richiesti da un'implementazione del SUAP.

Introduzione

Nel progettare una soluzione per il problema affrontato, è emerso che la gestione documentale viene a costituire il nucleo di ogni servizio interattivo progettato per il SUAP. Infatti, sia le normative contenute all'interno della base di conoscenza del SUAP, sia le domande uniche necessarie ad avviare il procedimento, sia le varie autorizzazioni e informative fornite sono documenti. A partire da questa constatazione, è stato deciso di sviluppare un framework per la gestione documentale che fosse il più generico possibile (in modo da favorirne il riuso) e che consentisse lo sviluppo di moduli che soddisfano i requisiti richiesti dal SUAP, illustrati nel secondo capitolo. Innanzitutto, come formato di rappresentazione dei documenti, si è scelto XML, che si sta avviando oramai a diventare uno standard “de facto” per la gestione documentale e che è stato scelto dal CNIPA[77] come formato di memorizzazione per le normative [CNIPA02]. A fronte di questa scelta, si è deciso di utilizzare un database XML nativo per la base di conoscenza documentale utilizzata dal framework, in modo da poter usufruire di tutti i vantaggi derivanti da XML come formato di rappresentazione. La scelta di progettare il servizio come web application, invece, è stata imposta come requisito necessario per garantire al cittadino una maggiore accessibilità allo Sportello Unico, dandogli come unico requisito per la fruizione del servizio la presenza di un browser sul

77 Centro Nazionale per l'Informatica nella Pubblica Amministrazione

suo sistema. Passando ad analizzare i requisiti del SUAP che riguardano la gestione documentale abbiamo individuato le funzionalità che devono essere presenti all'interno del framework:

- Capacità di content management: per gestire i rapporti interorganici con gli altri uffici della struttura comunale è necessario offrire la possibilità di creare, mantenere e pubblicare documenti di qualsiasi tipo (content management). Inoltre queste funzionalità sono richieste per la manutenzione e la pubblicazione della base di conoscenza normativa. Infine, considerato il fatto che il cittadino può utilizzare strumenti eterogenei per visualizzare le informazioni fornite dal SUAP (web browser, PDA, cellulari, etc), il framework deve consentire di pubblicare facilmente la medesima informazione su canali differenti (multi channel publishing).
- Capacità di knowledge management: la necessità, per quanto riguarda i rapporti con il cittadino o l'imprenditore, di navigare, consultare ed effettuare ricerche all'interno della base di conoscenza normativa, impone la presenza all'interno del framework di funzionalità di knowledge management. Queste funzionalità sono inoltre richieste in fase di costruzione della modulistica: in questa fase può infatti essere necessario effettuare delle ricerche all'interno della base di conoscenza per recuperare le componenti necessarie a costruire il modulo che verrà presentato all'utente.
- Modello di sicurezza: data la necessità di presentare lo Sportello Unico attraverso un'interfaccia facilmente accessibile dall'esterno, il framework deve fornire sia dei meccanismo per l'autenticazione dell'utente, sia un sistema di permessi sulle operazioni da svolgere all'interno della base di conoscenza.
- Funzionalità di editing interattivo: considerata la natura interattiva e basata su web della nostra soluzione, il framework deve fornire gli strumenti per creare, anche in maniera assistita, modulistica, richieste e documentazione in generale.

Content Management

Introduzione e definizione di Content Management

“Le buone idee sono, il più delle volte, facili da capire e, nascendo da necessità contingenti, sono pertanto facili da accettare. La parte difficile arriva quando le si devono trasformare in qualcosa di concreto. Un sistema di Content Management (CM), non fa eccezione: è una grande idea, è facile da capire ma è difficile da realizzare”[Kartchner98]. Un sistema di CM può essere definito come un framework per la creazione, la memorizzazione, l'accesso, la modifica, l'aggiornamento, il controllo e la visualizzazione dei contenuti digitali in vari formati, indipendentemente dal fatto che si trovino su Internet, in una intranet o in un sistema aziendale. In questo modo, il costo aggiuntivo di ogni modifica e del successivo aggiornamento dell'output si riducono drasticamente nel corso del tempo, portando così ad una riduzione dei costi sia in fase di produzione che in fase di manutenzione. Inoltre l'implementazione di un CM permette di migliorare le comunicazioni sia all'interno dell'ente che verso il cittadino o eventuali enti esterni.

Condizioni che permettono di sfruttare al meglio l'uso di un sistema di Content Management

- Possibilità di strutturare le informazioni: qualsiasi tipo di informazione deve potere essere organizzata in modo logico. Inoltre le componenti in cui l'informazione è strutturata devono rimanere valide indipendentemente dal contenuto e dalla struttura delle altre parti.
- Frequenti aggiornamenti: nel caso di informazioni che una volta pubblicate non devono più essere modificate, non ha alcun senso impiegare un sistema di CM. Viceversa, un CM si rivela particolarmente efficace quando viene usato per documenti che sono aggiornati con una certa frequenza, soprattutto nel caso in cui le modifiche riguardino solo una minima parte del documento

(ad esempio correzioni o piccoli aggiornamenti).

- Più persone partecipano alla stesura del documento: il sistema di CM realizzato deve offrire gli strumenti necessari per la gestione di documenti che derivano dalla collaborazione di più persone.
- I documenti hanno bisogno di essere pubblicati su più canali: i documenti gestiti dal sistema di CM devono essere visualizzati in un certo numero di formati differenti.

Per quello che riguarda il problema affrontato, queste condizioni sono presenti in diversa misura: la documentazione riguardante lo Sportello Unico è strutturata per sua stessa natura, essendo in molti casi la composizione di parti di altri documenti anch'essi strutturati. A proposito dell'aggiornamento frequente dei documenti, c'è da osservare che, mentre la parte normativa, ovviamente, viene raramente modificata una volta che è stata inserita, vi sono casi (come quello delle annotazioni) in cui può essere necessario un aggiornamento frequente. La collaborazione di più persone è necessaria in quanto uno stesso documento può poter essere modificato da un gruppo ristretto di utenti e infine, per quello che riguarda la pubblicazione su più canali, è necessario che i documenti pubblicati su web siano consistenti con quello che viene (eventualmente) stampato, presentando la necessità di una presentazione omogenea su due canali alquanto differenti.

Componenti di un sistema di Content Management[78]

Tipicamente i sistemi di CM differiscono tra di loro per le funzionalità che mettono a disposizione, tuttavia se ne possono identificare alcune che di solito sono presenti ovunque.

- Data Repository: struttura[79] che facilita l'accesso, l'aggiornamento e la ridistribuzione dei contenuti. La modalità secondo cui questa struttura viene organizzata è fortemente dipendente dal modo in

78 Questa classificazione è basata su [Kartchner98]

79 Generalmente si tratta di un database, ma può essere anche implementata in altro modo, ad esempio su filesystem

cui il contenuto deve essere accessibile. Il formato dell'informazione, solitamente è SGML, XML, oppure semplice testo codificato in ASCII o, preferibilmente, Unicode. È fondamentale che il repository sia reso accessibile sia dalla Intranet che da Internet e che quindi offra dei meccanismi di protezione che ne regolino l'accesso.

- Interfaccia utente: insieme di schermate che permettono all'utente di interagire con i dati. Un CM, tipicamente, è un insieme di prodotti dove verranno utilizzate interfacce differenti ma con lo stesso aspetto, specificatamente quelle dei browser o dei word processor.
- Workflow: un sistema di CM deve permettere di conoscere cosa stia accadendo ad un determinato componente del contenuto. Lo schema del workflow terrà quindi traccia di ogni elemento, della sua storia (check-in/check-out) e del suo versionamento. Tutte queste funzionalità permetteranno di controllare lo stato di un documento per sapere se è stato acceduto, se è stato modificato, se è stata richiesta una copia della versione modificata, se è stato rimandato all'autore per modificarlo o infine se sia stato accettato nella sua versione finale. Inoltre il versionamento potrà tenere traccia di quale versione del documento sia riferita (integrata) all'interno di altri documenti o di quale sia stata pubblicata. Infine lo schema del workflow dovrebbe essere in grado sia di generare dei rapporti personalizzabili nei quali siano fornite le informazioni relative allo stato dei documenti in vari formati, sia di permettere a chi pubblica i documenti di controllare e di tenere traccia del contenuto e dei processi coinvolti.
- Strumenti di stampa: perché un CM sia in grado di rendere disponibili i contenuti su più canali di distribuzione, è necessario che siano presenti dei filtri, capaci di prendere le informazioni presenti nel repository e di formattarle in uno dei vari formati. Ad esempio un filtro potrebbe generare la versione per il Web

(XHTML), un altro quella per la stampa (PDF) ed uno la versione ottimizzata per i palmari.

Multi-channel publishing

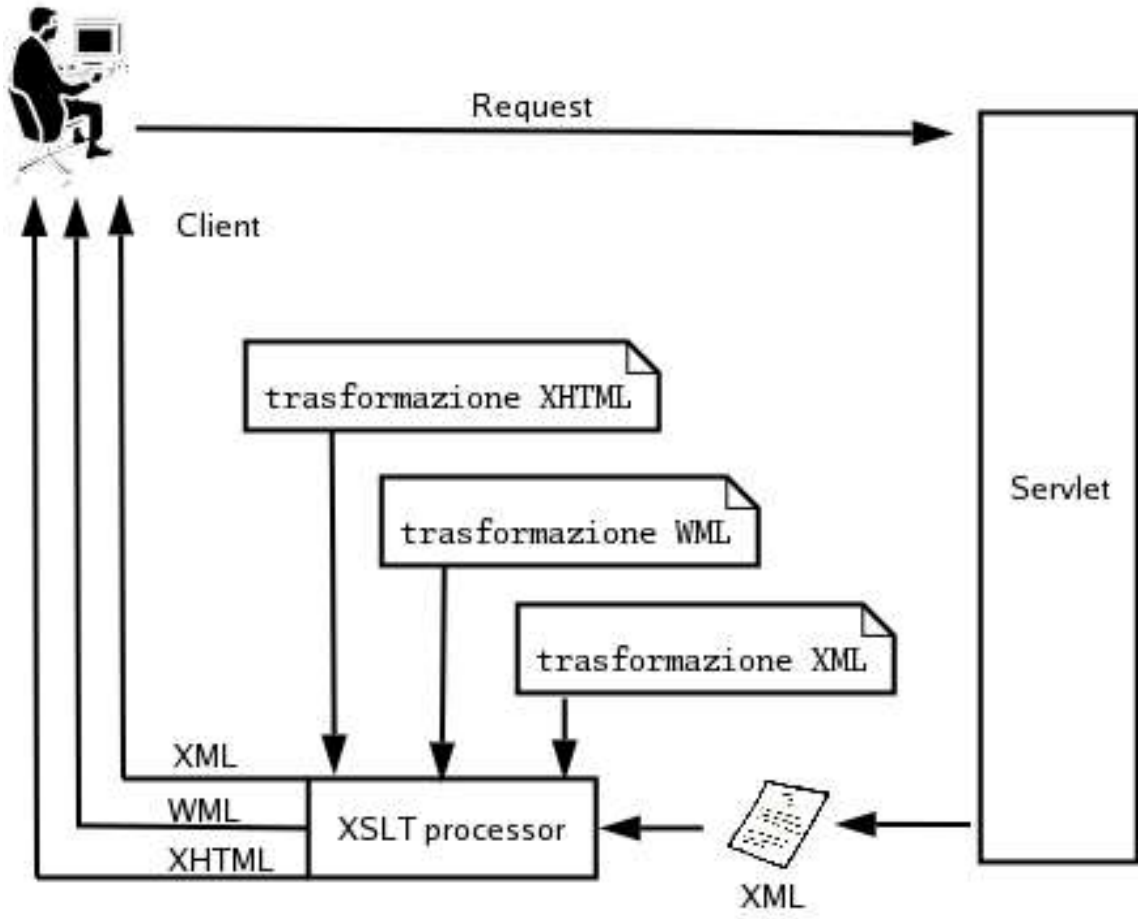

Figura 8: multi-channel publishing

Le pubbliche amministrazioni si trovano spesso nella necessità di dover raggiungere velocemente il maggior numero di persone possibili indipendentemente dal tipo di mezzi (PDA, cellulari, web browser, sintesi vocali) che vengono utilizzati per accedere ai contenuti pubblicati. Questo tipo di funzionalità viene definito in letteratura come "multi-channel publishing". Lo slogan usato spesso dai promotori di questi "sistemi di pubblicazione multicanale" è "Create Once, Publish Anywhere"[80], proprio per sottolineare che il

80 “Crea una sola volta, pubblica ovunque”

principale beneficio di questa funzionalità è la separazione tra contenuto e presentazione. Questo porta principalmente a due vantaggi: il primo è che qualsiasi modifica ai contenuti è immediatamente propagata su tutti i canali di comunicazione, mentre il secondo è che l'aggiunta di un nuovo canale di comunicazione richiede soltanto la creazione di un adeguato filtro di presentazione, evitando modifiche o duplicazioni dei contenuti.
La pubblicazione multicanale richiede che il processo di modifica dei contenuti avvenga indipendentemente dalla formattazione specifica dei media di presentazione, e dato che ognuno di questi ultimi necessita che i dati siano codificati in un formato differente, devono essere disponibili dei filtri di conversione per ciascuno di essi. Un altro requisito fondamentale è che ogni elemento che forma il contenuto da pubblicare debba essere memorizzato come un'unità logica discreta, e che queste unità logiche siano strutturate secondo uno schema ben definito, in modo da poter sviluppare dei filtri di conversione indipendentemente dal contenuto. Questo spiega perché gli standard XML/XSL siano così diffusi nel campo del multi-channel publishing. Queste due tecnologie, infatti, offrono una grande flessibilità, sia quando si tratta di applicare il contenuto al contesto, sia quando si deve visualizzare un documento secondo le convenzioni proprie di un particolare mezzo.
In definitiva, il requisito fondamentale per implementare un valido sistema di pubblicazione multicanale è avere tecnologie che permettano una separazione naturale tra la fase di authoring e quella di pubblicazione, in modo da pubblicare rapidamente un contenuto su un vasto numero di canali differenti.

Fase di authoring e groupware

Uno dei pregi maggiori dei sistemi di CM è sicuramente quello di non centralizzare la fase di authoring dei documenti, consentendo a più persone diverse, magari situate in locazioni fisiche differenti, di

contribuire al popolamento e al mantenimento della base documentale. Un esempio che può aiutare a capire meglio i vantaggi di un approccio di questo genere potrebbe essere una pubblica amministrazione in cui non solo i singoli individui possono mantenere i contenuti delle sezioni documentali a loro affidate, ma ogni ufficio o dipartimento deve anche poter gestire i contenuti di propria competenza assumendosene la responsabilità; inoltre i dipendenti delle varie aree devono poter lavorare sugli stessi contenuti eliminando le barriere tecnologiche e di sincronizzazione del lavoro che possono venire a crearsi. Quindi un'implementazione che consenta di gestire un approccio collaborativo all'authoring (groupware) diventa un requisito fondamentale per utilizzare appieno un sistema di CM. Questo approccio solitamente necessita di un formato standard dei dati in cui gli utenti inviano i documenti al modulo di pubblicazione, ma normalmente è consigliabile nascondere i dettagli del formato utilizzato dietro un'interfaccia familiare all'utente, come possono essere quella di un word processor oppure quella di una form HTML. Queste interfacce rappresentano soluzioni che stanno praticamente agli antipodi l'una dell'altra, dato che ognuna di esse si focalizza su aspetti differenti del problema. Dal punto di vista della presentazione e della facilità di utilizzo l'interfaccia di tipo word processor è certamente la migliore, ma dal punto di vista implementativo può essere molto complesso realizzarla. Inoltre, dato l'alto grado di libertà nella formattazione dei dati che di solito viene dato all'utente in questo tipo di interfaccia utente, può diventare molto complesso riuscire a isolare le diverse sottocomponenti del documento e, nella maggior parte dei casi, si è costretti a considerarlo come un blocco di dati monolitico che viene considerato solo nella sua interezza. Al contrario un'interfaccia realizzata attraverso form HTML è certamente più scarna, ma presenta tutta una serie di vantaggi che ne consigliano l'uso: innanzitutto un utente del sistema di groupware può inserire i suoi

documenti senza bisogno di alcun software particolare, se non un browser web e un sistema di connessione al server centrale, inoltre la suddivisione logica in varie sezioni data dai diversi campi di inserimento dati consente di partizionare logicamente il documento nelle sue sottocomponenti in maniera semplice ed immediata. Un ulteriore problema che caratterizza i sistemi di questo tipo, infine, consiste nel fatto che ogni utente deve poter avere accesso solo ai documenti di sua competenza, e questo implica l'implementazione di un qualche sistema di permessi.

Problemi legati all'implementazione di un CM

Sebbene, come abbiamo visto prima, i sistemi di CM offrano moltissimi vantaggi a coloro che li adottano, ci sono due grossi ostacoli che ne bloccano la diffusione su larga scala. Il primo è che attualmente il non utilizzarli non costa niente, ed è quindi facile ignorare gli eventuali risparmi futuri in favore di quelli che si ottengono nel breve termine, data anche la scarsità dei mezzi economici a disposizione della Pubblica Amministrazione per le infrastrutture informatiche[81]. Ad esempio, l'installazione e la configurazione di un sistema di CM può comportare costi particolarmente elevati nel breve periodo, e il ritorno dell'investimento non avrà luogo finché non sarà completamente integrato nel business process dell'organizzazione. A questo si deve aggiungere che coloro che decidono le strategie di un ente di piccole o medie dimensioni, difficilmente sono portati a pensare nel lungo periodo sia per motivi economici, sia per motivi culturali. Il secondo ostacolo consiste nel fatto che qualsiasi organizzazione, nel momento in cui decida di adottare un modello di business che si basi su una soluzione di CM, deve dipendere per un buon lasso di tempo dalle società che hanno realizzato questa soluzione e, nel caso i tempi di

81Come emerge anche da una recente ricerca Formez: si veda in proposito la tabella 2.30 di [Bevilacqua02].

consegna non vengano rispettati l'organizzazione si troverebbe nell'impossibilità di utilizzare la sua base documentale.

Knowledge Management

introduzione: il valore della conoscenza

In termini semplici, quello che comunemente viene chiamato "Knowledge Management" si può definire come “Un insieme di processi volti a creare, catturare, memorizzare, condividere, applicare e riutilizzare la conoscenza”[Sydanmaanlakka00]. In un'era in cui la disponibilità di informazioni cresce molto rapidamente, grazie alla sempre maggiore disponibilità di personal computer e di accessi ad Internet, il vero problema non è più la carenza di informazioni, ma piuttosto la necessità di riconoscere i dati veramente importanti in mezzo ad un oceano fatto di rumore di fondo. Si prenda ad esempio l'intero corpus giuridico della legge italiana: la semplice accessibilità dei documenti non basta per potersi servire di una base di conoscenza così vasta. È necessario organizzare i dati secondo un certo schema, indicizzarli e renderne agevole la navigazione, e questo è qualcosa in cui l'informatica riesce attraverso l'uso delle ultime tecnologie. Non bisogna però pensare che si possano ottenere gratuitamente risultati accettabili. Al contrario, è necessario progettare attentamente il sistema di organizzazione della conoscenza, pensando sia al sistema informatico utilizzato che all'utente finale. Quest'ultimo, infatti, deve riuscire facilmente ad accedere alla base di conoscenza contenuta nel sistema, ed è quindi fondamentale progettare un'interfaccia che gli consenta di farlo.

Il punto di vista dell'utente

La parola chiave, in questo contesto è "usabilità". Un sistema di Knowledge Management deve essere utilizzabile dagli utenti con facilità, altrimenti la sua utilità diventa nulla. Ci sono un certo numero di attributi che sono normalmente associati all'usabilità, e

anche se queste qualità sono importanti in qualsiasi sistema informatico, diventano assolutamente fondamentali in un contesto di e-government, dove gli utenti (i cittadini) hanno spesso una conoscenza approssimativa del dominio (la pubblica amministrazione). Date queste premesse, si possono individuare alcuni requisiti critici del sistema per i quali deve essere spesa una buona parte di risorse in fase di progettazione. Per prima cosa è necessario un sistema di navigazione tra i dati della base di conoscenza che guidi l'utilizzatore, attraverso passaggi guidati, fino all'informazione di cui ha bisogno. Questo, però, non è sufficiente nel caso che l'utente abbia a disposizione solo alcune "parole chiave" significative di quello che vuole cercare; per queste evenienze, un sistema di ricerca sulla base di conoscenza diventa un altro dei requisiti critici del sistema. Infine, la scarsa conoscenza del dominio porterà sicuramente ad un aumento degli errori in fase di inserimento dati, per cui un robusto e flessibile sottosistema di validazione dell'input finisce col diventare di importanza critica. Questi sottosistemi, per loro natura, sono pesantemente knowledge-based, piuttosto che content-based, cioè si basano non tanto sul contenuto specifico dell'informazione, ma piuttosto sul significato che viene dato a quell'informazione; e anche se questo sembrerebbe essere falso per il sottosistema di ricerca, va considerato che un'implementazione non banale non si limita a ricerche “grezze” su tutta la base di conoscenza, ma consente di restringere il campo di ricerca alle sue parti significative. In breve, per avere un sistema di Knowledge Management usabile, il contenuto non è abbastanza: c'è bisogno di qualcos'altro.

Non solo contenuti: il problema della semantica

Come si è già detto, l'informazione, per essere utilizzata agevolmente sia da un sistema informativo, sia da un essere umano, non può rimanere nella forma “grezza” in cui viene trovata, ma deve venire

raffinata ed elaborata in vista dell'utilizzo che se ne deve fare. Il primo passo di questo processo, di solito, è l'acquisizione in forma digitale e la successiva memorizzazione in una memoria di massa da cui potrà venire recuperata quando ce ne sarà bisogno. Questa fase è resa estremamente delicata da un problema che molto spesso viene sottovalutato: l'informazione cartacea od orale possiede, oltre al contenuto vero e proprio, anche tutta una serie di informazioni prammatiche (contenute, ad esempio, nella formattazione del testo, nel corpo dei caratteri, nell'intonazione della voce o in una conoscenza a priori posseduta da chi consulta l'informazione) che vengono spesso perse nella fase di digitalizzazione. Inoltre bisogna considerare che spesso c'è solo un piccolo sottoinsieme di queste informazioni che ha una reale importanza: ad esempio un sistema di consultazione delle leggi dello Stato ha bisogno dei riferimenti incrociati tra i vari documenti, in modo da poter navigare con facilità con un sistema ipertestuale tra un decreto e le leggi collegate, mentre un sistema di stampa dei certificati (che deve includere le parti rilevanti delle leggi che lo riguardano) ha piuttosto bisogno di informazioni riguardanti la struttura del singolo documento, come la divisione in capi, sezioni, articoli, commi e paragrafi, per poterne utilizzare solo le parti rilevanti. È quindi importante decidere, prima di acquisire l'informazione, quali informazioni semantiche si debbano includere e (di conseguenza) quale formalismo si debba utilizzare per memorizzarle; una scelta potrebbe essere, ad esempio, quella di decidere se immergere la semantica all'interno del contenuto (la strada utilizzata normalmente per le informazioni testuali attraverso l'utilizzo di tag) oppure mantenerla in una sezione a parte (come nel caso di informazioni contenute in immagini grafiche, che sono difficilmente editabili per questi scopi). Un esempio di formalismo che comprende sia informazioni semantiche, sia contenuto vero e proprio è dato da XML, uno standard di cui si è parlato più in dettaglio nel capitolo 4 e che abbiamo utilizzato proprio per il fatto di

strutturare l'informazione aggiungendo informazioni semantiche al contenuto.

Descrizione del Framework

Architettura generale

Le funzionalità del framework sono state suddivise in varie componenti, secondo lo schema illustrato in figura 9:

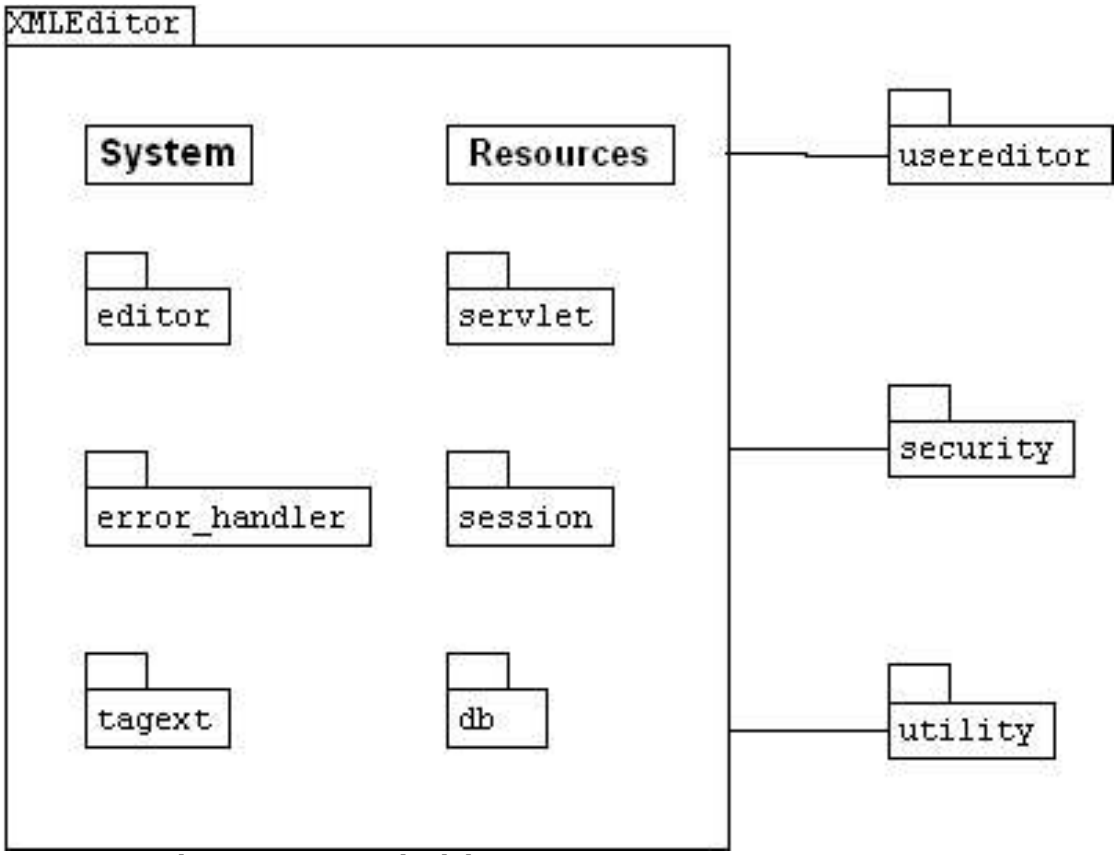

Figura 9: architettura generale del sistema

XMLEditor è il modulo centrale del framework e comprende le funzionalità principali per la gestione documentale, che sono state raggruppate nei seguenti sottomoduli:

- db: all'interno di questo modulo sono presenti le funzionalità di content e knowledge management relative alla gestione della base di conoscenza, cioè la possibilità di creare, modificare, recuperare ed eliminare documenti XML di qualsiasi tipo.

- Editor: all'interno di questo modulo sono presenti le funzionalità di content management che riguardano l'editing e la verifica della correttezza e dell'integrità all'interno del singolo documento XML.
- Error_handler: all'interno di questo modulo sono fornite le funzionalità di gestione delle situazioni di errore che possono verificarsi durante l'utilizzo dell'applicazione.
- Servlet: all'interno di questo modulo sono realizzate le servlet di interfaccia verso il client esterno.
- Session: in questo modulo viene gestito lo stato della web application.
- Tagext: in questo modulo sono contenuti i custom tag equivalenti alle servlet di interfaccia verso il client esterno.
- Interfaccia con il sistema (System): questa classe è necessaria per astrarre le funzionalità del sistema sottostante.
- Risorse statiche (Resources): le istanze di questa classe contengono tutte le risorse statiche necessarie all'applicazione (come ad esempio le classi di trasformazione dei documenti o la locazione dei repository).

Tutte le funzionalità relative all'autenticazione degli utenti e alla loro gestione sono state inserite all'interno del modulo “usereditor”. Questo modulo contiene, oltre alle procedure necessarie ad autenticare un utente ed effettuare le operazioni di login e logout, anche il codice necessario a creare, modificare ed eliminare gli utenti. Queste ultime funzionalità, però, non sono state implementate autonomamente, ma fanno un uso intensivo delle funzionalità offerte dal resto del framework, visto che i dati relativi agli utenti vengono memorizzati come documenti[82].

Il modulo “security”, invece, contiene tutte le funzionalità necessarie a risolvere i permessi di accesso delle applicazioni che utilizzano il framework, attraverso la gestione di una Access Control List.

82 Per un ulteriore livello di sicurezza, la password è memorizzata all'interno del documento in forma crittata.

Infine, il modulo “utility” contiene tutta una serie di funzioni di utilità non specifiche del framework, come le funzionalità di logging degli errori e dei warning, la gestione delle espressioni regolari o le funzioni di crittografia. Queste funzionalità sono state isolate dal resto del framework per favorirne un eventuale riuso in altri progetti.

Gestione della base di conoscenza

Le caratteristiche che una base di conoscenza deve possedere per essere utilizzata da un sistema informatico di gestione del SUAP sono diverse:

- Documentale: la base di conoscenza deve essere basata sulle entità gestite dal SUAP, cioè sui documenti.
- Distribuita: i documenti della base di conoscenza non devono necessariamente essere collocati all'interno di un singolo database fisico. Ad esempio le normative regionali potrebbero risiedere in una banca dati comune a tutta la regione, mentre le varie normative comunali potrebbero risiedere in banche dati locali ad ogni comune.
- Eterogenea: a seconda del loro tipo, i documenti possono essere memorizzati all'interno di database con caratteristiche differenti. Ad esempio la vecchia documentazione non ancora convertita in XML potrebbe risiedere all'interno di una struttura basata sul filesystem, oppure potrebbe essere memorizzata all'interno di un database relazionale.

Il modulo “db” del framework contiene tutte le funzionalità necessarie a creare, modificare, recuperare ed eliminare documenti XML da una base di conoscenza che possiede le caratteristiche elencate. Il fatto che la base di conoscenza possa essere distribuita o eterogenea ha portato ad eliminare la soluzione di creare un'unica classe concreta che realizzasse queste funzionalità per tutti i documenti. Al contrario, si è progettata un'interfaccia (CollectionManager) da specializzare separatamente per ogni tipo di

documento. Ogni implementazione dell'interfaccia conosce il tipo di documento da gestire ed il tipo di repository su cui il documento è memorizzato, consentendo all'applicazione di utilizzare in maniera trasparente l'interfaccia senza preoccuparsi della topologia e della tipologia dei repository (come mostrato in figura 10).

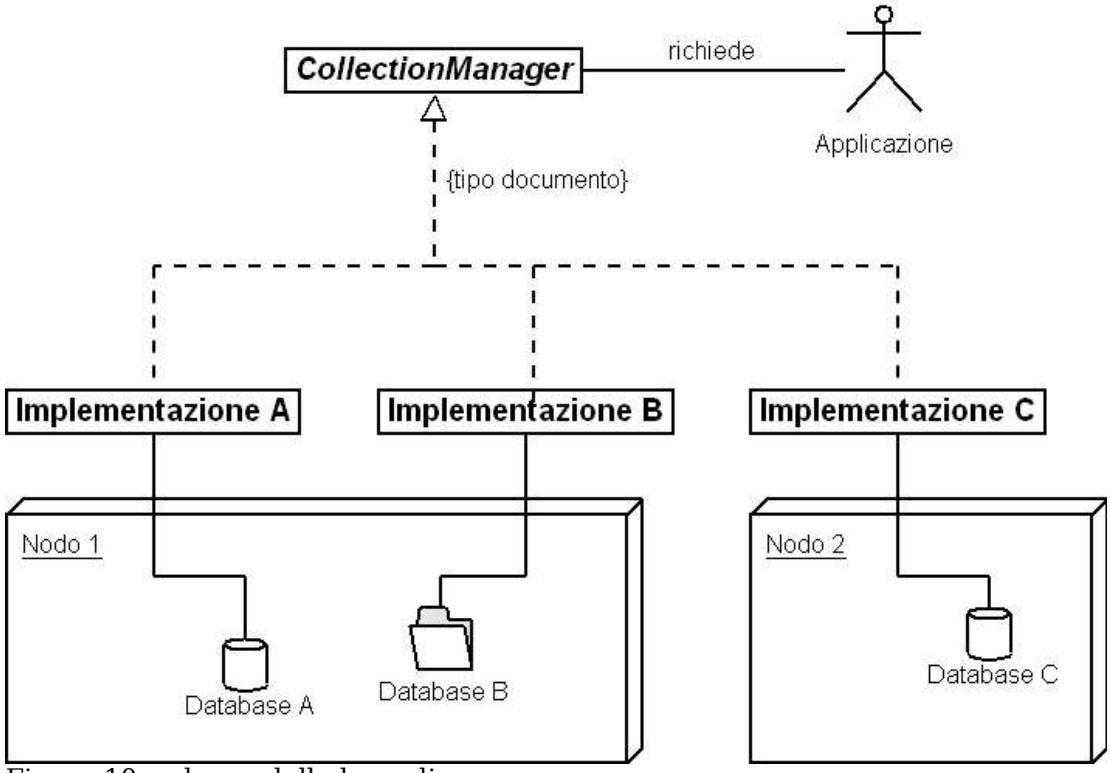

Figura 10: schema della base di conoscenza

La corrispondenza tra tipo di documento e implementazione da utilizzare è demandata alla classe di gestione delle risorse dinamiche che verrà descritta nel paragrafo dedicato alla gestione delle risorse. Questa soluzione offre quindi la flessibilità necessaria a gestire una base di conoscenza documentale distribuita ed eterogenea; visto, però, che per la maggior parte dei tipi di documento presenti in un SUAP le operazioni di gestione documentale sono realizzate alla stessa maniera, il framework offre anche due implementazioni base di CollectionManager, che forniscono metodi generici applicabili alla

maggior parte dei documenti. La scelta di gestire i vari tipi di documento in questo modo ha permesso non solo di nascondere la struttura fisica della base di conoscenza, ma anche di gestire dei “tipi virtuali” di documento. Infatti, tra i documenti che possono essere presentati dal SUAP, ve ne sono alcuni che non esistono fisicamente all'interno della base di conoscenza, ma vengono creati aggregando parti di altri documenti. Utilizzando delle implementazioni di CollectionManager specifiche per questo tipo di documenti, essi possono essere gestiti dal resto del framework e dalle applicazioni basate su di esso, esattamente allo stesso modo di quelli fisicamente presenti all'interno della base di conoscenza.

Modifica dei documenti

All'interno del modulo “editor” sono presenti le funzionalità che semplificano la compilazione assistita dei documenti. Questa caratteristica del framework si rende particolarmente necessaria in un servizio interattivo per la pubblica amministrazione, dato che i dati inseriti andranno poi a formare un documento con valore legale. Inoltre, il fatto che la compilazione dei documenti possa venire effettuata da cittadini senza conoscenze specifiche, rende ancora più importante una funzionalità di verifica dei documenti compilati. Le funzionalità di compilazione assistita, fondamentalmente, sono tre:

- inizializzazione del documento: in questa fase il documento recuperato dalla base di conoscenza viene trasformato per semplificarne l'editing: ad esempio, se un elemento definito come opzionale dal XML-Schema non è presente nel documento recuperato dal repository, il framework si preoccuperà di aggiungere un elemento vuoto al documento, in modo che all'utilizzatore venga presentato un campo vuoto in corrispondenza dell'elemento considerato, che può essere compilato o lasciato vuoto.
- Verifica del documento: in questa fase il documento modificato dall'utilizzatore viene validato attraverso il suo XML-Schema, e

possono venire effettuati tutti i controlli sul contenuto degli elementi ritenuti necessari.

- Finalizzazione del documento: in questa fase il documento modificato viene preparato per essere salvato all'interno della base di conoscenza, eliminando tutti gli elementi vuoti definiti come opzionali dal XML-Schema e inserendo eventuali elementi aggiuntivi, calcolati a partire dai dati forniti dall'utente[83]. All'interno di questo modulo è stata anche inserita la classe che contiene il documento modificato, il suo stato attuale e tutte le informazioni necessarie all'editing durante la sessione di modifica.

Gestione degli errori

All'interno del modulo “error_handler” si trovano tutte le funzionalità di gestione degli errori non recuperabili. La scelta di consentire alle applicazioni che utilizzano il framework di potere definire una visualizzazione personalizzata degli errori non recuperabili è stata dettata dalla necessità di integrare la presentazione dei messaggi di errore con la web application. Inoltre questo consente di fornire all'utente un descrizione dell'errore verificatosi che sia significativa anche per chi non possiede conoscenze tecniche di livello avanzato. Ad esempio, tra queste informazioni possono essere inserite quelle necessarie a contattare l'assistenza tecnica.

Interfaccia con il client

I moduli che si occupano dell'interfaccia con il client esterno sono “servlet”, “tagext” e “session”. I primi due forniscono le interfacce vere e proprie verso l'utente, ed hanno funzionalità equivalenti; l'unica differenza è che, mentre le servlet vengono utilizzate autonomamente, i custom tag contenuti in tagext sono utilizzati all'interno di Java Server Pages (JSPs). In questo modo il framework fornisce la possibilità di utilizzare entrambi i metodi di

83 Un esempio pratico può essere la conversione della data inserita dall'utente dal formato “gg/mm/aa”ad un formato di rappresentazione interno che faciliti le operazioni di ricerca per data.

presentazione. Sia le servlet che i custom tag si occupano solo dell'aspetto di controllo dell'applicazione, invocando le funzionalità del framework a seconda dei parametri passati dal client e dello stato attuale della web application. Questo stato è mantenuto e gestito dalle classi presenti nel modulo “session”, visto che il protocollo HTTP non fornisce un meccanismo nativo per implementare uno stato.

Gestione delle risorse

Dato che una delle linee guida di progettazione implicava l'estendibilità e l'adattabilità del framework a diverse situazioni, la maggior parte dei moduli sono stati resi indipendenti, per quanto possibile, dai dettagli del sistema ospite. In questa prospettiva tutte le funzionalità di interfacciamento con i moduli di sistema (come il database, il filesystem o i mail server) sono state concentrate in un'unica interfaccia che è stata chiamata “System”. Questo consente di centralizzare i cambiamenti da apportare qualora si dovesse utilizzare, ad esempio, un'implementazione differente del database o un sistema di notifica agli utenti differente dal protocollo SMTP utilizzato per le mail. Per quello che riguarda i servizi forniti dalla System, sono stati divisi in due parti, ognuna delle quali è gestita da una classe differente, che viene fornita dalla System su richiesta del codice applicativo:

1. Resources: fornisce l'accesso alle risorse statiche del sistema (documenti di configurazione, fogli di stile, nomi di classi).
2. DBConnection: fornisce l'accesso alle risorse dinamiche del sistema (principalmente l'accesso ad uno strato di persistenza come può essere un database o un filesystem)

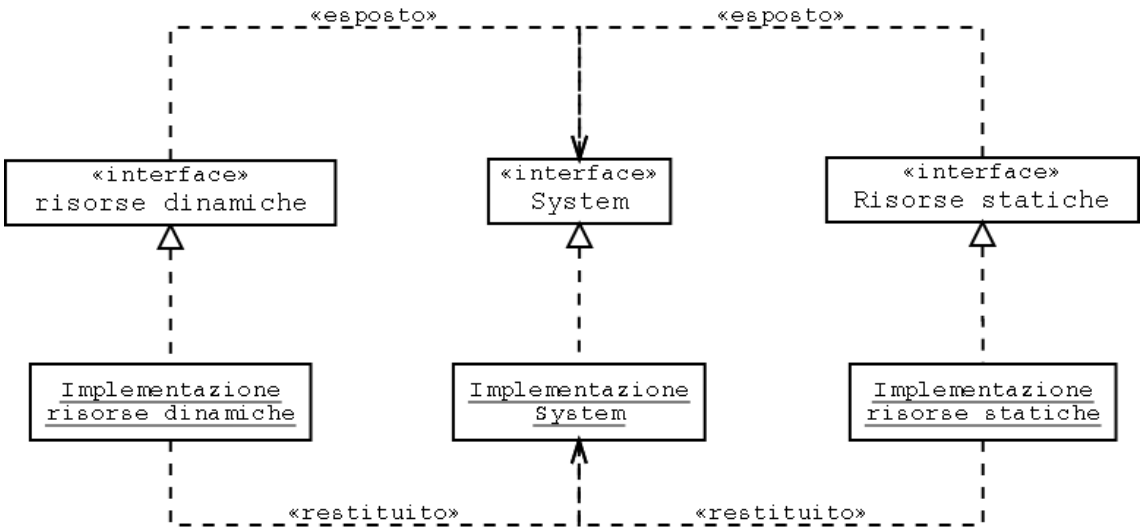

Figura 11: struttura della System

Come si può notare dal diagramma delle classi illustrato in figura 11, ci sono due livelli isomorfi tra loro: quello dell'implementazione e quello delle interfacce. Il framework vede solamente quest'ultimo, consentendo di effettuare con un minimo sforzo cambiamenti alla configurazione del sistema che normalmente avrebbero un costo molto alto. Un esempio che può far capire l'utilità di questa scelta di progettazione può essere questo: si supponga di dover cambiare il tipo di database dove è memorizzata la base documentale da XML nativo a SQL; normalmente questi cambiamenti influirebbero sulla struttura dell'intero framework, ma con la struttura utilizzata c'è solo bisogno di creare un'altra implementazione dell'interfaccia “risorse dinamiche” che implementi le modifiche necessarie, fare un cambiamento al modulo System dell'applicazione in modo che restituisca un'istanza dell'implementazione sviluppata, e il software viene aggiornato alle nuove specifiche in maniera quasi indolore e, soprattutto, localizzata[84]. Oltre a fornire le astrazioni di interfacciamento col sistema, la System gestisce tutti quegli aspetti che dipendono fortemente dall'ambiente hardware/software sul quale

84 In caso di malfunzionamenti a seguito dell'aggiornamento, è chiaro che gli errori possono essere solo nel modulo modificato. In questo modo il campo di ricerca degli errori viene drasticamente ridimensionato

è installata l'applicazione, come può essere la gestione di un pool di risorse da assegnare ai vari utenti del servizio, che varia a seconda del carico che la piattaforma di installazione può sopportare senza aumentare troppo i tempi di risposta.

Risorse dinamiche (DBConnection)

Questa classe, inserita all'interno del modulo “db”, fornisce l'interfaccia tra il sistema e i repository delle risorse create dinamicamente. Ogni risorsa di questo tipo è individuata da un sistema di coordinate univoco. Si è scelto di utilizzare come sistema di coordinate una coppia di stringhe, la prima delle quali indica il tipo di risorsa, mentre la seconda è un identificativo univoco. La scelta di utilizzare una coppia di chiavi invece di una singola chiave è stata suggerita dal fatto che differenti tipi di risorse possono essere situate in differenti locazioni fisiche della base di conoscenza e possono essere recuperate in modi diversi. Passando il tipo come parametro esplicito, la scelta degli algoritmi di gestione della risorsa può essere realizzata in maniera efficiente. Le coordinate sono state definite come stringhe perché, anche se meno compatte di un valore numerico, hanno il vantaggio di poter essere passate senza conversioni di sorta come parametro di chiamata di una servlet o di una JSP, attraverso la loro concatenazione nell'URL di chiamata.

Risorse statiche (Resources)

La classe che si occupa di gestire le risorse statiche, contiene al suo interno quelle risorse e quei parametri che consentono di personalizzare l'applicazione. Ad esempio le locazioni fisiche dei database sui quali è distribuita la base di conoscenza, i fogli di stile XSL che forniscono la presentazione o i messaggi di errore utilizzati dal modulo “error_handler”. Il principale beneficio offerto da questa classe è quello di semplificare la personalizzazione dell'applicazione sviluppata utilizzando il framework, concentrando tutte le informazioni di localizzazione in un unico punto. In questo modo una

singola applicazione può essere facilmente adattata ai diversi contesti presenti all'interno delle pubbliche amministrazioni.

Utilizzo del framework

Presentazione dei documenti

La presentazione dei documenti avviene in modo molto semplice: al modulo di presentazione vengono passate le coordinate del documento da visualizzare e un identificatore della modalità di presentazione desiderata. In base a quest'ultimo e al tipo di documento da visualizzare, viene selezionato un foglio di stile XSLT dal gestore delle risorse statiche, mentre attraverso il gestore delle risorse dinamiche si recupera il documento corrispondente alle coordinate passate. A questo punto si utilizzano le API di JAXP per trasformare il documento in un frammento XHTML attraverso il foglio di stile ed infine si invia il risultato ottenuto al client. Questo modulo esiste in due versioni, come servlet e come “extension tag”, ma la struttura di entrambi è fondamentalmente la stessa. Bisogna notare che la notevole semplicità del paradigma utilizzato non limita affatto l'utilizzo del modulo a casi poco complessi: in effetti quello che è stato fatto è stato di demandare la gestione della presentazione (che può essere anche molto complicata) al foglio di stile utilizzato. Tale spostamento di complessità, però, non è privo di vantaggi, dato che in questo modo per la presentazione si utilizza un linguaggio come XSLT, che è specializzato per operare trasformazioni di documenti, al posto di un linguaggio general-purpose quale può essere Java. Oltre al modulo principale di presentazione, che restituisce pagine XHTML, è presente anche un modulo di stampa che utilizza lo stesso paradigma appena descritto. Le uniche differenze sono che:

- Non esiste una versione implementata come extension tag di questo modulo, dato che non restituendo codice visualizzabile a schermo è inutile inserirlo all'interno di una pagina.

- Il foglio di stile produce un documento XSL-FO invece di un XHTML, ed il risultato ottenuto viene trasformato in PDF (o in un altro formato supportato da FOP, come ad esempio Postscript) per poter essere stampato.

Come si è visto in quest'ultimo caso, quindi, il paradigma utilizzato per questo modulo rende la pubblicazione multi-canale molto semplice: quando c'è bisogno di visualizzare il documento su dei nuovi media, basta scrivere dei nuovi fogli di stile che trasformino il documento in un formato adatto al medium scelto.

Editing dei documenti

Uno dei moduli centrali del framework è sicuramente quello che gestisce l'editing dei documenti. Vista la complessità e la quantità di requisiti richiesti da questo modulo, si è scelto di realizzarlo in maniera leggermente differente degli altri moduli. La servlet che gestisce le varie fasi dell'editing, infatti, implementa un automa a stati finiti, descritto in figura 12,in cui il passaggio di stato viene effettuato richiamando la servlet attraverso un link interattivo attivato dall'utente oppure attraverso una redirezione automatica.

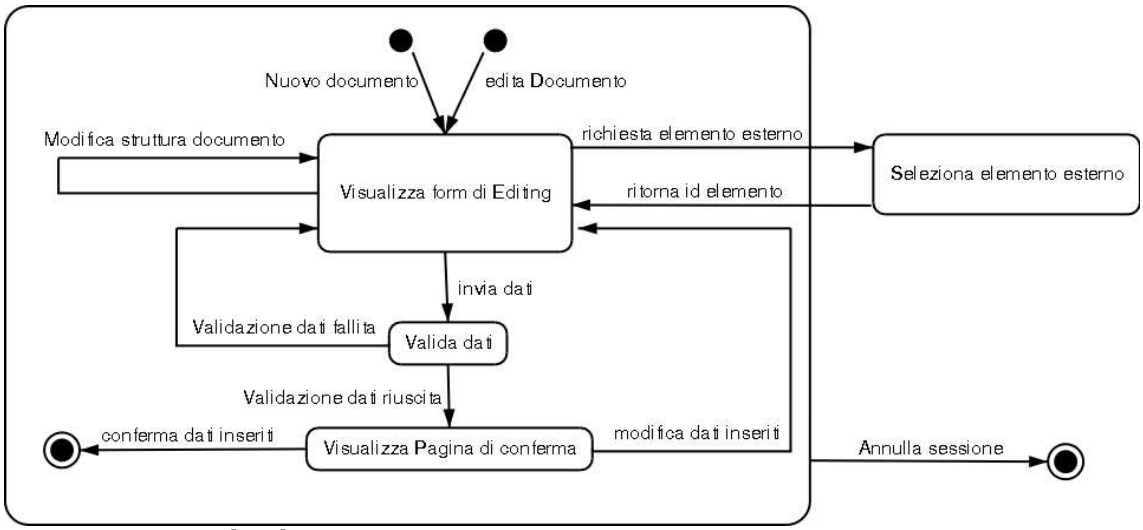

Figura 12: stati di editing

Per ogni tipo di documento che si vuole editare è necessario definire una classe che implementi la logica delle operazioni di modifica (come ad esempio la validazione o l'inserimento automatico dei contenuti nel documento) attraverso un'interfaccia ben definita. In questo modo l'unica modifica sostanziale da apportare al framework per adattarlo all'editing di un ulteriore tipo di documento, è quello di sviluppare una nuova classe che implementi quell'interfaccia e inserirla nel sistema. Oltre a questo, durante tutta la durata dell'operazione di modifica, viene mantenuto un oggetto di sessione[85] in cui sono contenuti i dati comuni a tutte le fasi del processo di modifica. In particolare contiene:

- il documento XML editato;
- un campo che indica se quest'ultimo è stato validato ed è risultato corretto;
- le coordinate che identificano il documento;
- lo stato attuale dell'automa.

Per l'editing, infine, è stata utilizzata una tecnologia proprietaria di Metaware che consente sia il riempimento automatico delle maschere di modifica con i dati del documento, sia l'aggiornamento automatico del documento con i valori inseriti nei campi modificati dall'utente.

Esaminiamo adesso più in dettaglio le varie fasi dell'editing:

1. Editing di un documento esistente: Questa è uno dei due modi in cui si può iniziare ad editare un documento:

   - La servlet riceve come parametri le coordinate del documento da modificare e l'identificatore del foglio di stile XSLT che contiene le informazioni di presentazione.

   - Attraverso queste informazioni, il framework consente di operare una trasformazione sul documento; infatti è possibile che il

85 Gli oggetti di sessione sono istanze di classi Java che gli application server conformi alle specifiche J2EE 2.x mantengono in vita per tutta la durata della sessione HTTP relativa ad un utente. Lo scopo di questi oggetti è quello di aggiungere uno stato ad un protocollo (HTTP) che ne è sprovvisto.

documento XML da modificare non contenga alcuni o tutti i campi definiti come opzionali dal suo XML-Schema. In questo caso è necessario espandere il documento inserendo le parti mancanti e lasciandole vuote, così che l'utente possa eventualmente riempirle.

- La servlet richiede alla DBConnection il documento ed inserisce il contenuto di quest'ultimo nelle maschere di inserimento dati della pagina visualizzata.
- In aggiunta a questa operazione si possono inoltre riempire automaticamente i campi del documento che è possibile calcolare a priori.

Un esempio di quest'ultimo punto potrebbe essere quello di un campo contenente il numero totale di copie degli allegati ad una domanda, calcolato automaticamente attraverso la somma del numero di copie di ogni singolo allegato.

2. Creazione di un nuovo documento: L'altro modo di iniziare ad editare un documento è quello di crearne uno completamente nuovo. In questo caso alla servlet non viene passato l'identificativo di un oggetto, ma solamente il tipo dell'oggetto da creare e il foglio di stile necessario per la visualizzazione, oltre ad altri parametri che possono essere necessari per specificare come deve essere creato il documento. A partire dal tipo ed eventualmente da altri parametri passati, il framework si preoccupa di fornire un identificativo univoco per il documento appena creato, dopo di che si procede in maniera analoga all'editing di un documento esistente. A parte questo, vale la pena di spendere due parole sul modo in cui viene creato il nuovo documento: utilizzando il fatto che per ogni tipo di documento è definito un XML-Schema[86], si è sfruttata la potenza del linguaggio XSLT scrivendo una template che utilizza le informazioni contenute nell'XML dello Schema, in

86 Nel caso il documento disponesse solo di una DTD, è possibile scrivere un tool che converta da DTD a XML-Schema, ed in effetti durante la realizzazione del framework un tool simile è stato sviluppato per ottenere lo Schema di alcuni vecchi tipi di documento per cui era fornita solamente la DTD

modo da ottenere come risultato un documento vuoto pronto per essere riempito dall'utente.

3. Visualizzazione maschera di inserimento dati: in questa fase la servlet applica semplicemente il foglio di stile indicato dal parametro ricevuto e invia la pagina ottenuta al client che l'aveva richiesta.
4. Modifica della struttura del documento: Spesso può accadere che all'interno di un documento, un nodo o un sottoalbero non abbiano cardinalità fissa; ad esempio un ipotetico XML che descrivesse una persona potrebbe contenere zero o più numeri di telefono presso cui contattarla. Per gestire questi casi è necessario prevedere la possibilità che l'utente possa aggiungere o togliere determinati campi, modificando la struttura (e non solo il contenuto) del documento editato. Ogni cambiamento strutturale viene richiesto direttamente richiamando la servlet con determinati parametri: questo fa sì che la modifica venga riportata direttamente sul documento contenuto nell'oggetto di sessione e che subito dopo venga visualizzata la form aggiornata applicando normalmente il foglio di stile all'XML.
5. Richiesta di un documento esterno: Un requisito che si è voluto implementare all'interno del framework è quello di poter riempire dei campi selezionando un documento “esterno” da cui recuperare i dati necessari. Un esempio può essere quello di dover inserire nel documento i dati del responsabile di un determinato ufficio: la cosa più semplice per l'utente è quella di navigare nell'applicazione che gestisce gli uffici, selezionandone uno e lasciando che il sistema riempa automaticamente i campi relativi. Per ottenere questo, semplicemente si sospende la sessione di editing e si passa il controllo ad un altro contesto che fornisce la gestione degli oggetti che si vogliono utilizzare[87]. Una volta terminata la selezione del

87 Nell'esempio riportato potrebbe esistere un'applicazione per la gestione degli uffici, indipendente da quella utilizzata, che viene sfruttata per la navigazione della lista di uffici.

documento, il controllo viene di nuovo passato alla servlet di editing, comunicando l'identificativo dell'oggetto selezionato in modo da permettere l'aggiornamento del documento.

6. Verifica della correttezza dei dati: Per evitare di appesantire la logica della parte di editing con pagine e pagine di codice di controllo (che oltretutto andrebbe riscritto per ogni tipo di documento e che aumenterebbe in maniera significativa la probabilità di inserire errori nel sistema), tutte le informazioni sui test da compiere per un determinato tipo di documento sono state inserite in un XML che viene processato da una apposita classe Java, che esegue automaticamente tutte le verifiche indicate nel file di configurazione. Grazie alla capacità di Java di utilizzare la reflection API per richiamare dinamicamente i metodi delle classi, però, questa scelta non ha fatto perdere né in generalità, né in flessibilità. Infatti ogni singola verifica contenuta nel documento di configurazione contiene due informazioni: la prima è una query XPath che identifica i nodi o gli attributi da verificare all'interno del documento, mentre la seconda è il nome del metodo che viene chiamato per verificare il contenuto di quei particolari nodi. Per ogni campo errato che viene individuato durante il processo, viene inserito all'interno del campo uno speciale tag di errore, contenente un messaggio relativo all'errore riscontrato. Alla fine del processo di verifica, se è stato trovato almeno un errore, lo stato dell'automa ritorna nella fase di editing. Gli errori vengono visualizzati dal foglio di stile della maschera di inserimento dati, che è stato progettato per individuare questi tag e stampare il messaggio di errore contenuto al loro interno vicino al campo con i dati non corretti.
7. Conferma modifiche: in questa fase vengono visualizzate tutte le informazioni inserite dall'utente, e viene data la possibilità di ritornare all'editing o di confermare i dati inseriti. In caso di conferma il sistema provvede a preparare il documento per essere

aggiornato fisicamente (ad esempio vengono eliminati eventuali tag opzionali rimasti vuoti)

8. Annulla sessione di editing: In ogni momento è possibile annullare tutta l'operazione di editing. In questo caso vengono perse tutte le informazioni inserite dall'utente.

Durante la fase di editing o creazione di un documento c'è la possibilità che l'utente debba creare o modificare un altro documento: un esempio potrebbe essere il caso in cui il documento debba contenere il riferimento ad un ente di appartenenza e quest'ultimo non sia compreso nella lista degli enti da cui effettuare la scelta. L'utente, in questo caso, deve poter creare un nuovo ente per inserirlo nel documento principale(vedi figura 13).

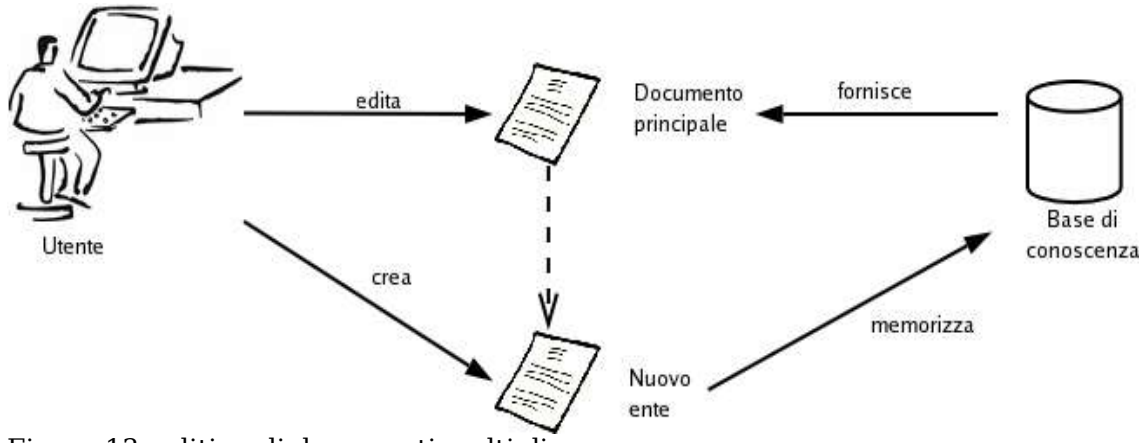

Figura 13: editing di documenti multipli

La creazione del nuovo ente, per ragioni di usabilità, non deve costringere l'utente ad abbandonare l'editing del documento, ma deve consentire di abbandonare temporaneamente le operazioni sul documento principale, creare o editare il documento secondario e ritornare all'editing una volta terminata l'operazione. Per avere la massima generalità, inoltre, la quantità di operazioni di editing che possono essere nidificate non deve avere un limite superiore fissato a priori. Dato che tutte le informazioni relative all'editing sono contenute in un unico oggetto di sessione, la soluzione adottata per

risolvere il problema consiste nella realizzazione di una struttura che può memorizzare un numero non limitato di stati di editing secondo un meccanismo FIFO. In questo modo, ogni volta che l'utente ha bisogno di editare un documento accessorio, il documento corrente viene salvato nella pila e, una volta terminata l'operazione, lo stato salvato è recuperato dalla testa della pila. Oltre a queste funzionalità deve essere prevista una funzionalità di svuotamento della pila, nel caso che l'utente decida di abbandonare l'intera operazione di editing.

Gestione del modello di sicurezza

La gestione della sicurezza nel framework realizzato è stata progettata in modo da consentire di definire sia permessi con una granularità molto fine, sia permessi che abbracciano un insieme più vasto di casi. Gli elementi di base che vengono utilizzati per costruire un modello di sicurezza sono le azioni e le regole. Le azioni sono insiemi di coppie (attributo, valore) che indicano le singole operazioni compiute sul framework. Un esempio di azione potrebbe essere questo insieme: [(id,”X00123”), (comando,”aggiorna”), (tipo,”normativa”)], in cui si indica l'operazione di aggiornare la normativa con identificatore “X00123”. Le regole, invece, sono oggetti Java che verificano una determinata condizione, come ad esempio che un utente abbia un determinato ruolo oppure che il documento sia stato editato almeno una volta. Ovviamente è impossibile basare un modello di sicurezza realistico sulle singole azioni, dato che bisognerebbe specificarne una per ogni documento. Per ovviare a questo problema, quelle che vengono utilizzate nella costruzione del modello di sicurezza per la singola operazione non sono delle azioni singole, ma piuttosto delle “template” formate da coppie attributo/espressione regolare. Ad esempio la template [(id,”X.”), (comando,”aggiorna|leggi”), (tipo,”normative”)] indica l'azione di aggiornamento o di lettura di tutte le normative il cui identificatore inizia con “X”. Template e regole, poi, vanno a formare

i “permessi”: un permesso consiste in una template e in un insieme di regole. Perché un permesso possa essere considerato valido rispetto ad un'azione, bisogna che quest'ultima corrisponda alla template e che le regole siano soddisfatte. Normalmente il permesso viene ritenuto valido se almeno una delle regole è soddisfatta ma, per garantire una maggiore flessibilità, questo comportamento può essere modificato attraverso l'assegnazione esplicita di due attributi della singola regola:

1. Final: se questo attributo è impostato e la regola è soddisfatta, allora il permesso è valido, a prescindere dalle altre regole
2. Critical: se questo attributo è impostato e la regola non è soddisfatta, allora il permesso non è valido a prescindere dalle altre regole

Per costruire un modello di sicurezza per un'applicazione, quindi, basta definire un insieme di permessi e riunirli all'interno di una lista di controllo degli accessi (ACL). La lista è memorizzata in un oggetto Java, che viene inizializzato all'avvio dell'applicazione attraverso la lettura di un file XML di inizializzazione e che, come tutte le risorse statiche, è contenuto all'interno di Resources. Per stabilire se un utente può eseguire una determinata azione in un certo momento, il framework richiede la ACL a Resources e cerca i permessi validi per compiere quell'operazione. Se non ne viene trovato nessuno, viene negata la possibilità di compiere l'azione in questione. Sopra a questo strato è stato costruito un meccanismo di gestione dei ruoli degli utenti, che fornisce degli insiemi di permessi, ognuno dei quali è identificato da un nome significativo. Tale meccanismo si rende necessario per fornire dei ruoli “standard” che evitano di specificare ogni volta i permessi in dettaglio.

Navigazione della base di conoscenza

All'interno del framework, l'esplorazione della base di conoscenza da parte dell'utente può avvenire in due modi. Uno di questi si appoggia su una struttura ad albero, in cui i documenti occupano le foglie,

mentre i nodi non terminali sono occupati da descrittori di ambito, che categorizzano la base di conoscenza secondo lo schema illustrato in figura 14

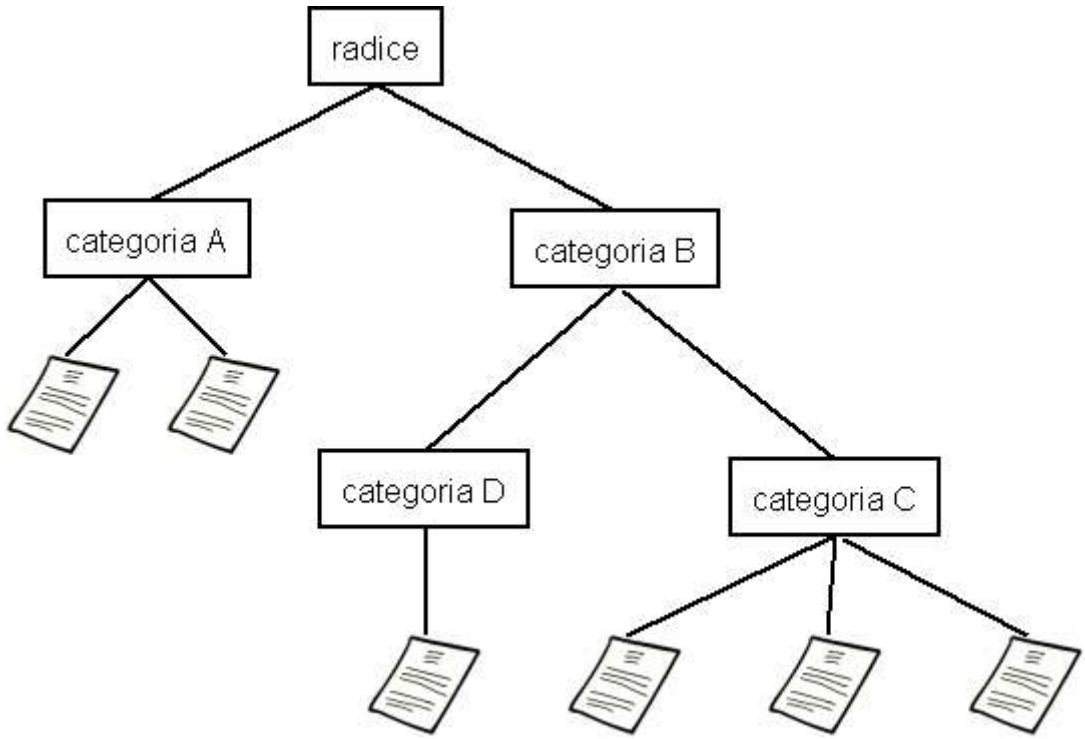

Figura 14: schema di navigazione

Questi descrittori di ambito, sono stati modellati anch'essi come documenti e vengono gestiti all'interno della base di conoscenza attraverso le funzionalità offerte dal framework. Inoltre, ogni documento e ogni descrittore di ambito contengono un riferimento all'ambito padre. Quando l'utente seleziona una categoria, all'interno del client, viene visualizzata la lista dei riferimenti ai documenti o alle categorie figlie. Questa lista non è altro che un documento di tipo “virtuale”, costruito a partire dalle intestazioni di tutti i documenti presenti nella base di conoscenza che hanno la categoria selezionata come padre, e che viene visualizzato come una lista di collegamenti. In questo modo, attraverso i riferimenti contenuti nelle intestazioni, l'utente può navigare dalla radice verso le foglie, selezionando via via le categorie che gli interessano. La navigazione

in senso contrario avviene invece mantenendo la storia delle categorie selezionate e visualizzandola in forma di lista di collegamenti. In questo modo l'utente non è vincolato a risalire l'albero di un nodo alla volta, ma può saltare ad uno qualsiasi degli antenati dell'elemento corrente (vedi figura 15)

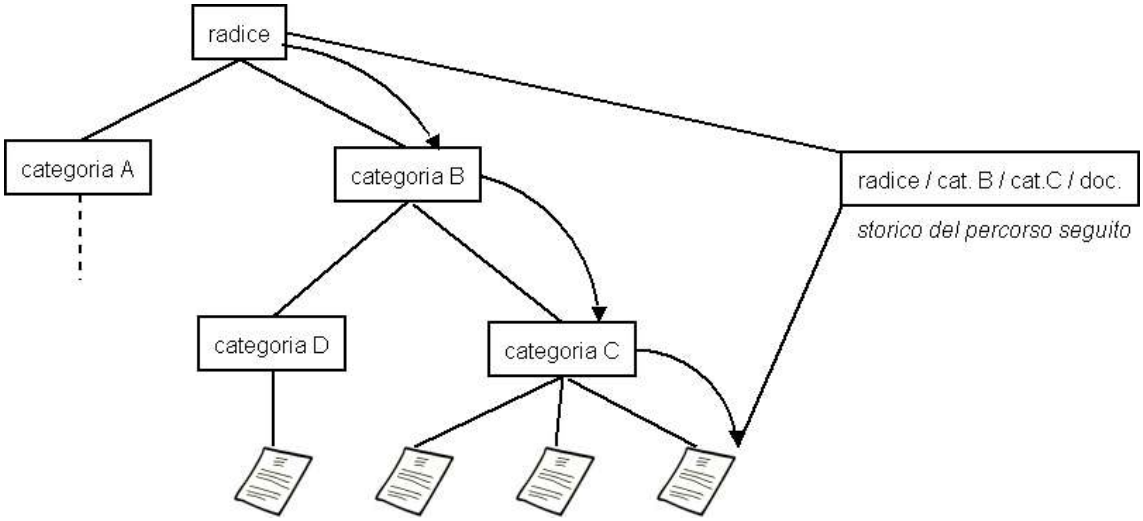

Figura 15: percorso di navigazione

L'altra modalità di navigazione avviene in maniera trasversale, attraverso i riferimenti ad altri documenti presenti nel documento considerato. Ad esempio, un'attività produttiva contiene i riferimenti agli endoprocedimenti necessari per attuarla. Questi riferimenti non sono altro che gli identificatori univoci dei documenti presenti all'interno della base di conoscenza, e questo consente di recuperare il documento riferito attraverso una semplice interrogazione sulla base di conoscenza.

Ricerca sulla base di conoscenza

La funzionalità di ricerca è stata implementata in modo da consentire una più rapida consultazione della base di conoscenza, specie nei casi in cui quest'ultima assume dimensioni talmente grandi che un esame manuale di tutti i documenti sarebbe improponibile. I requisiti della funzione di ricerca sono che:

1. deve essere possibile individuare qualsiasi documento che

contenga un determinato frammento di testo al suo interno.

2. La ricerca non deve essere dipendente dal fatto che le lettere del frammento di testo cercato siano maiuscole o minuscole.
3. La ricerca può essere limitata ad un singolo tipo di documento
4. I risultati della ricerca non devono essere i documenti interi, ma solo delle intestazioni. Da ognuna di queste intestazioni deve essere possibile ricavare le informazioni per recuperare il documento a cui appartiene.
5. Deve essere possibile definire come criterio di ricerca la presenza di diversi frammenti di testo, sia in “and” che in “or” tra di loro.

Per realizzare una funzionalità che soddisfacesse tutti questi criteri, si sono utilizzate le caratteristiche di pattern matching del linguaggio XPath, che hanno estremamente facilitato la scrittura della funzionalità. Tra le funzioni standard definite nella specifica di XPath, infatti, figura la funzione booleana “contains()”, che prende come argomento due stringhe, ed è vera se e solo se la prima stringa passata contiene al suo interno la seconda. Sfruttando il fatto che un qualsiasi documento XML può venire trasformato in una stringa[88], attraverso questa funzione si può sapere se un determinato frammento di testo è presente in un documento. Per realizzare ricerche complesse, che implicano più frammenti di testo, è necessario innanzitutto definire una sintassi che consenta all'utente di esprimerle e che possa essere ricordata facilmente: nel nostro caso si è scelto di utilizzare una rappresentazione in forma normale disgiunta, in cui i termini sono i frammenti di testo oggetto della ricerca, gli “and” logici sono rappresentati dal carattere '+', mentre gli “or” logici sono rappresentati da uno spazio. Per evitare di confondere l'utilizzatore si è scelto di non inserire la possibilità di negare i termini, ma modificare il modulo in modo da prevedere una tale possibilità è banale. Un esempio della sintassi utilizzata

88 È da notare che questa stringa non contiene i nomi dei tag, ma solo il contenuto testuale dei nodi del sottoalbero. Questo rende possibile la ricerca evitando di effettuare il matching con i nomi dei tag.

potrebbe essere “agricoltura+edilizia allevamento+edilizia”, che indica tutti i documenti che contengono la parola “edilizia” e che contengono “agricoltura” oppure “allevamento”. Per realizzare la query XPath corrispondente alla ricerca voluta, però, questa stringa non è sufficiente: un'altra informazione necessaria è il tipo di documento che deve essere recuperato, in modo da restringere la ricerca solo al dominio desiderato (per fare una ricerca in più domini è sufficiente combinare più ricerche, una per dominio, e fondere i risultati ottenuti). Ad esempio, la stringa precedente, applicata al dominio degli enti, dà origine alla query XPath “ente[(contains (current(),”agricoltura”) and contains(current(), “edilizia”)) or (contains(current(),”allevamento”) and contains(current(),”edilizia”))] ”, dove la funzione current() indica il nodo corrente. Per quello che riguarda il requisito dell'indipendenza dalle maiuscole e dalle minuscole, è stato necessario aggirare la povertà di funzioni fornita dallo standard XPath in questo modo: ogni volta che viene fatto un confronto, tutte le lettere minuscole vengono trasformate in maiuscole attraverso la funzione “translate”, che converte tutte le occorrenze di una lettera in un'altra. Ovviamente questa conversione comporta una certa perdita di efficenza, ma questo requisito è stato ritenuto fondamentale per l'usabilità del servizio: in un ambiente come quello della pubblica amministrazione, in cui non esistono standard ben definiti per l'utilizzo di maiuscole e minuscole, un tipo di ricerca *case-sensitive* avrebbe un'utilità scarsamente rilevante. Gli altri requisiti, invece, sono stati soddisfatti con uno sforzo minimo, grazie alla scelta di utilizzare XML-DB come database: per quello che riguarda la possibilità di limitare la ricerca solo ad alcuni tipi, il database è stato organizzato inserendo tutti gli elementi dello stesso tipo in una singola collection, in modo da rendere la ricerca più efficiente. Il recupero dei soli header dei documenti, invece, è stato reso banale dalle informazioni semantiche fornite dai tag XML, dato che è bastato racchiudere la parte di documento che contiene

l'intestazione all'interno di un tag apposito e recuperare solo quella parte.

Utilizzo del framework per la realizzazione di un SUAP

Il framework sviluppato consente di informatizzare la gestione documentale di un SUAP in tutti i suoi aspetti.

Base di conoscenza normativa

La base di conoscenza normativa può essere creata e mantenuta attraverso le capacità di content management del framework, memorizzando le normative relative al SUAP come documenti XML su una base di conoscenza che può essere anche distribuita. La transizione da una eventuale base di conoscenza precedente che utilizzava un altro formato per la gestione documentale può essere risolta creando un CollectionManager particolare che si interfaccia con i vecchi documenti. Questa, comunque, deve essere considerata solo una soluzione di transizione, visto che le caratteristiche avanzate del framework (come la navigazione o la creazione di nuovi documenti a partire da frammenti di altri documenti) sono possibili solo se la documentazione è memorizzata in formato XML. La navigazione della base di conoscenza avviene secondo una suddivisione in categorie (“materie”) e le singole normative possono essere visualizzate nel client o renderizzate in PDF con il meccanismo di presentazione fornito dal framework, che fornisce le funzionalità di pubblicazione multicanale del documento attraverso l'uso dei fogli di stile XSL.

Gestione degli utenti

Le informazioni relative agli utenti vengono memorizzate all'interno della base di conoscenza come documenti, utilizzando le funzionalità di gestione documentale del framework. Le autorizzazioni concesse agli utenti vengono definite da un insieme di ruoli predefiniti (“utente anonimo”, “utente registrato”, “amministratore”,

“redattore”, etc.) ad ognuno dei quali è associato un insieme di permessi secondo il modello di sicurezza descritto nei paragrafi precedenti. Il problema della creazione dell'utente iniziale che ha i diritti per creare altri utenti viene risolto durante la fase di installazione e configurazione dell'applicazione, attraverso un tool a linea di comando, indipendente dalla web application, che è in grado di creare un utente che possiede le autorizzazioni necessarie. Una volta creata una base minima di utenti per la gestione del SUAP, questo utente “di servizio” viene rimosso.

Creazione di una domanda unica

La creazione di una domanda unica avviene componendo diversi frammenti di documento contenuti nella base di conoscenza (endoprocedimenti, attività produttive, normative, etc.) con informazioni fornite dall'utente per formare un unico documento. L'utente può aggiungere questi frammenti navigando sulla base di conoscenza attraverso le funzionalità offerte dal framework. Oltre a questo, viene data la possibilità di creare un modulo per la dichiarazione antimafia, attraverso una form XHTML nella quale si possono inserire i dati necessari.

Tavolo tecnico virtuale, news e agenda

Come già specificato all'interno del secondo capitolo, le modalità di comunicazione interorganica all'interno del SUAP possono avvenire secondo diverse modalità. Innanzitutto gli operatori possono collegare dei documenti particolari (“note”) a tutti i documenti presenti sulla base di conoscenza. I riferimenti a queste annotazioni vengono visualizzati interrogando la base di conoscenza per recuperare tutte le note che hanno un riferimento al documento considerato. La garanzia che le note vengano mostrate esclusivamente agli operatori è ottenuta impostando in modo opportuno i permessi relativi all'azione di visualizzazione delle note. Un'ulteriore modalità di comunicazione interorganica avviene

attraverso un sistema di pubblicazione degli appuntamenti. Questi ultimi sono modellati come documenti inseriti all'interno della base di conoscenza e mantenuti attraverso le funzionalità offerte dal framework; inoltre la loro visualizzazione avviene attraverso delle interrogazioni basate sulle coordinate temporali presenti all'interno di ogni appuntamento. Ad esempio la web application può visualizzare gli appuntamenti dei prossimi 15 giorni oppure quelli previsti tra il primo e l'ultimo giorno del mese successivo. La gestione delle news, infine, viene realizzata in modo molto simile a quella degli appuntamenti, con la differenza che le coordinate temporali considerate si riferiscono alla data di pubblicazione della news[89] piuttosto che alla data dell'appuntamento.

Rilascio dell'atto finale

La costruzione del documento relativo all'atto finale, può essere realizzata selezionando, attraverso le funzionalità del framework, i frammenti necessari provenienti dai diversi documenti che contengono i responsi forniti dagli enti terzi responsabili del processo autorizzativo. Questo documento può venire renderizzato in PDF, attraverso le funzionalità offerte dal framework, per essere stampato e firmato dai responsabili di sportello.

89 Questa data viene inserita automaticamente dal modulo editor nella fase di finalizzazione dell'editing del documento.

Capitolo 6: Un esempio concreto: la comunicazione di avvio

In questo capitolo verrà illustrato un esempio di applicazione che utilizza il nostro framework, nella quale verrà realizzata la fase di comunicazione di avvio del procedimento unico per le attività produttive. In questa descrizione verrà evidenziata l'ampia flessibilità del framework ed inoltre verranno descritte in dettaglio le varie fasi della creazione di un'applicazione specifica

Introduzione

Si è scelto di presentare in maniera più dettagliata la fase del procedimento unico riguardante la Comunicazione di Avvio perchè al suo interno vengono utilizzate la maggior parte delle funzionalità offerte dal framework.

Tra i compiti che devono essere svolti dal responsabile del SUAP, c'è quello di preparare una lettera che avverta l'imprenditore, che ha compilato e inviato una Domanda Unica e del fatto che sia stata inoltrata la richiesta di attivazione dell'attività produttiva da lui richiesta. La compilazione di questo documento richiede l'immissione di sei tipi di informazioni:

- dati provenienti dal sistema (workflow) quali il numero di protocollo e la data della Comunicazione di Avvio;
- l'elenco dei destinatari ai quali si deve inviare questa lettera;
- dati provenienti dalla relativa Domanda Unica (DU) come: il numero della DU, il numero del protocollo e la data, il tipo di procedimento unico, il tipo di richiesta, il tipo di attività produttiva, gli endoprocedimenti coinvolti;
- informazioni relative al personale del comune coinvolto in questa comunicazione come il referente della pratica, il responsabile del trattamento dei dati personali e il responsabile del procedimento (firmatario) di cui può essere indicato l'orario di ricevimento;
- informazioni locali al Comune come gli orari di apertura al pubblico, l'indirizzo, l'e-mail ed il sito web;

- data e luogo della compilazione della Comunicazione di Avvio

Come si può osservare, la maggior parte dei dati presenti nella Comunicazione di Avvio o proviene da altri documenti oppure può essere calcolata. Sono quindi davvero pochissime le informazioni conosciute soltanto dall'operatore. L'applicazione Comunicazione di Avvio mette in risalto due peculiarità del nostro framework: la prima è che sfruttando la base di conoscenza e i documenti ivi contenuti, è in grado di creare un nuovo documento dove i campi da immettere interattivamente sono semplici e relativamente pochi, la seconda è che, riducendo l'apporto da parte dell'operatore umano, che adesso ha il ruolo di collaboratore alla stesura del documento, diminuisce le possibilità che si possa verificare un errore nei dati immessi.

Passaggio da documento cartaceo alla DTD

Il primo passo che abbiamo compiuto per realizzare questa applicazione, è stato quello di conoscere il dominio in cui sarebbe stata impiegata, per valutare correttamente la semantica dei dati contenuti nella versione cartacea della Comunicazione di Avvio. Ci siamo quindi preoccupati di capire quali informazioni provengano da altri documenti, quali siano statiche, quali dipendano dal Comune e quali vadano immesse interattivamente. Con queste informazioni siamo stati in grado di scrivere la DTD che descrive la struttura della Comunicazione di Avvio. Infine abbiamo utilizzato il tool da noi sviluppato per convertite la DTD in un XML-Schema che sarebbe poi stato utilizzato dall'applicazione per la creazione del documento minimale.

Creazione delle collection

Una volta creato l'XML-Schema, il passo successivo è stato quello di capire quante collection fisiche e virtuali ci sarebbero servite. Basandoci sulle linee guida per lo sviluppo di un'applicazione basata sul nostro framework, ci siamo resi conto che avevamo bisogno di due collection fisiche (con i relativi gestori) e di una collection

virtuale. Vediamo nei dettagli i motivi di questa scelta:

- La collection fisica per le domande uniche si è rivelata necessaria per disporre di un meccanismo indipendente dal gestore delle Comunicazioni di Avvio con il quale accedere alle informazioni significative per il loro recupero come il nome del tag che ne contiene l'id univoco all'interno di tutta la collection; il tag che punta all'intestazione e quello che restituisce il puntatore all'inizio del documento.[90]
- La collection fisica per le Comunicazioni di Avvio, oltre che per motivi analoghi alla precedente, ci è servita per ridefinire il metodo che crea un nuovo oggetto di una collection, visto che ad una nuova Comunicazione di Avvio si devono aggiungere delle informazioni come quelle relative alla domanda unica a cui si riferisce o quelle relative al Comune presso cui è installata l'applicazione.
- La collection virtuale è stata invece necessaria per presentare la lista di tutte le domande uniche e delle relative comunicazioni di avvio che sono presenti nel database, eventualmente visualizzandole in base ad un certo intervallo temporale. Per come è strutturato il framework, si dovrebbe potere fissare una coppia di valori (tipo, identificatore) che restituisca il documento a cui corrisponde questa lista, ma ovviamente questo documento non esiste fisicamente sul database, perchè è il risultato di due query XPath eseguite su due collection differenti. Per risolvere questo problema, visto che il framework permette di associare ad ogni collection (indicata dal tipo) una query che restituisca un particolare documento della collection stessa, abbiamo pensato di creare una collection virtuale che contenga un solo elemento, al cui interno si trovano i risultati delle query eseguite sulla collection delle Domanda Unica e su quella delle Comunicazione di

90 Non è previsto che le domande uniche si possano creare o editare da questa applicazione in quanto abbiamo assunto che esista un'altra applicazione specifica per tale compito.

Avvio. La realizzazione di questo documento virtuale, non è stata però immediata perchè ci siamo trovati a dovere fondere i risultati delle query eseguite sulle due collection: risultati che per il paradigma del nostro framework devono essere restituiti attraverso un flusso SAX per poi essere utilizzati per la fase di visualizzazione. Per fonderli abbiamo sfruttato il fatto che nell'analisi dei documenti XML tramite SAX, quando viene trovato il tag che identifica la fine di un documento, viene segnalato un evento particolare che indica, appunto, la fine del documento. Questo evento viene intercettato dal framework, facendo in modo che quando si arriva alla fine del documento, venga eseguita l'altra query, i cui risultati sono sempre restituiti in un flusso SAX. Soltanto quando si arriva al fine di quest'ultimo documento vengono eseguite anche le operazioni di chiusura relative al primo. Questo procedimento può essere ripetuto in maniera iterativa: se avessimo avuto bisogno di creare una collection virtuale per gestire le informazioni di (ad esempio) tre collection invece che di due, come è successo in questo caso, sarebbe stato sufficiente utilizzare questa tecnica una terza volta.

Schemi di presentazione per la visualizzazione

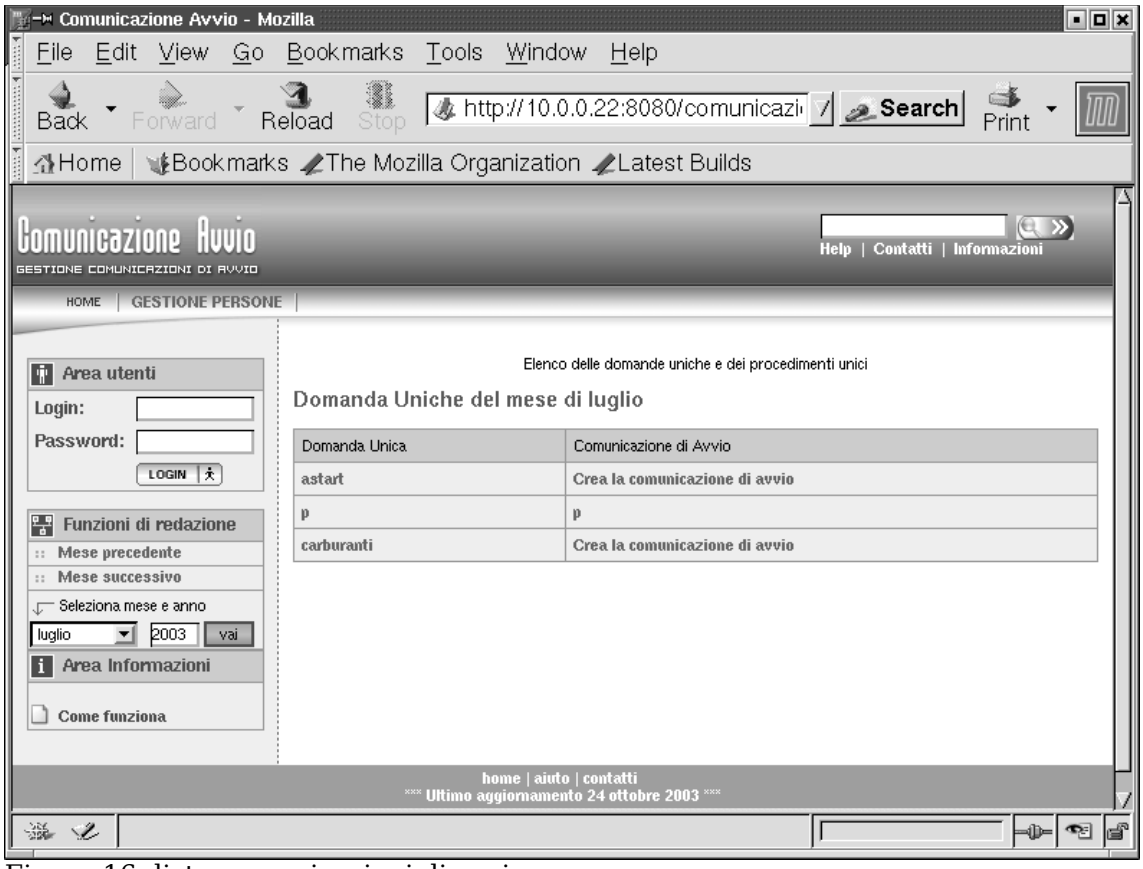

Figura 16: lista comunicazioni di avvio

La fase successiva è stata quella di progettare i fogli di stile necessari per visualizzare rispettivamente la lista con l'elenco delle domande uniche e delle Comunicazioni di Avvio, i dettagli della singole domande uniche ed i dettagli delle singole Comunicazioni di Avvio. Da notare che, per come è stata costruita la collection virtuale della Comunicazione di Avvio, ovvero disponendo di un documento al cui interno si trovano sia gli identificatori delle domande uniche che gli identificatori delle Comunicazioni di Avvio con il numero della domanda unica a cui fanno riferimento, siamo in grado visualizzare le domande uniche che sono state pubblicate in un determinato arco di tempo con visualizzata a fianco la Comunicazione di Avvio, se è stata compilata oppure un collegamento che dà la possibilità di

compilare la relativa Comunicazione di Avvio. Per capire se sia stata o meno inviata la Comunicazione di Avvio di una determinata domanda unica, è stato sufficiente prendere i vari elementi della collection virtuale e controllare se esisteva anche un identificativo per la Comunicazione di Avvio.

Schemi di presentazione per l'editing

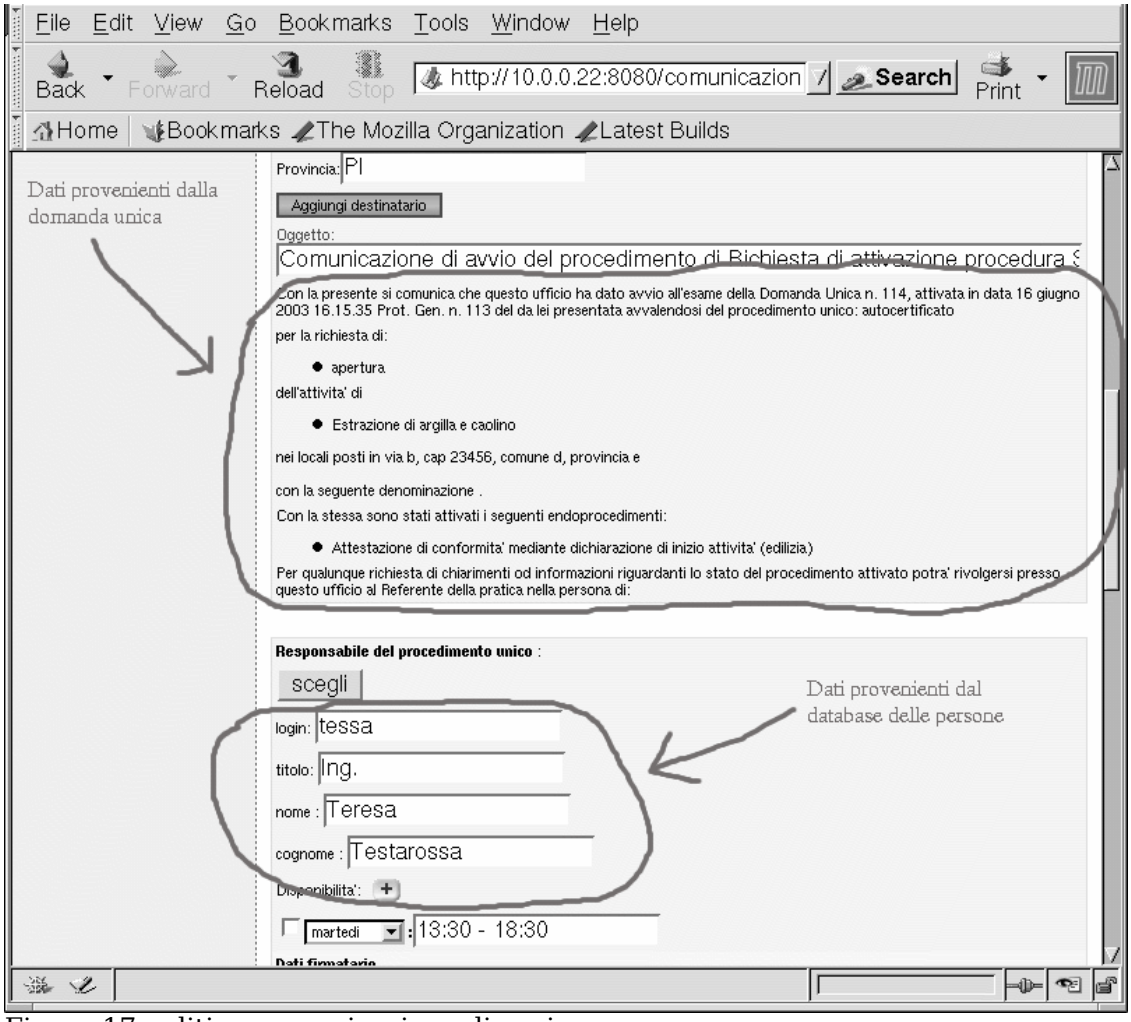

Figura 17: editing comunicazione di avvio

Questa è senza dubbio la parte più interessante di tutta l'applicazione perché vengono sfruttate buona parte delle funzionalità messe a disposizione del nostro framework: quando viene creato un nuovo documento, questi viene arricchito con

informazioni provenienti da altri documenti presenti nella base di conoscenza, oppure da documenti memorizzati sul filesystem. Quando viene creata una nuova Comunicazione di Avvio, al documento minimale vengono aggiunte alcune informazioni prima che questi venga presentato all'utente. Scendendo nei dettagli, queste informazioni sono:

- Tutti i dati relativi alla domanda unica di cui si sta compilando la relativa Comunicazione di Avvio.
- Le informazioni statiche relative al comune e ai dipendenti che sono coinvolti nella domanda unica, che si trovano in un file XML.
- Le informazioni provenienti dal workflow (Ci è stato fornito soltanto il formato di questo parametri, ma non come accedervi. Pertanto abbiamo deciso che siano editabili dall'operatore)

Sebbene la struttura della domanda unica da noi utilizzata sia stata fornita da Metaware, un eventuale cambiamento della struttura stessa, impatterebbe in maniera poco significativa sui cambiamenti da apportare all'applicazione, visto che il caricamento dei dati è stato centralizzato in unico metodo, che sarebbe il solo a dover essere modificato, mentre la struttura della Comunicazione di Avvio resterebbe completamente invariata.

Scrittura di un validatore

Una volta che sia stata terminata la fase di editing di un documento, si può passare alla sua validazione ovvero a controllare che: i campi obbligatori siano stati correttamente compilati e che quelli opzionali, nel caso in cui non siano vuoti, contengano dei valori corretti. In questa fase, sono coinvolte tre componenti: la prima è un file XML che contiene la lista dei controlli da eseguire su di un determinato insieme di nodi; la seconda è una classe che contiene i metodi per eseguire i controlli; la terza è un file di configurazione che indica per ogni tipo di errore quale file XML, contenente il relativo messaggio, debba essere caricato nel documento. Vediamo nei dettagli come

funziona il processo di verifica: una volta che l'utente ha finito di compilare la form, come abbiamo visto nel diagramma degli stati mostrato nel capitolo cinque, questo documento viene inviato alla EditObject che richiederà all'editor specifico di quella collection di documenti di controllare il documento. Prima che abbia inizio la validazione vera e propria, vengono però eseguite due operazioni: la prima è quella di rimuovere gli eventuali tag di errore che possono essere presenti nel documento se non è la prima volta che ne viene richiesta la verifica. Successivamente viene caricato il file XML che contiene una lista di triple che sono rispettivamente il nodo su cui effettuare il controllo, il metodo che esegue il controllo e un valore di verità che indica se questo controllo sia opzionale. Vediamo due esempi di come sono fatte queste triple:

1° caso:

```
<tag select="cap">
        <check>is5Digits</check>
</tag>
```

Il significato di questa tripla è di controllare, sull'insieme dei nodi di tipo ELEMENT il cui nome del tag è cap, se il contenuto del figlio nodo di tipo PCDATA siano cinque numeri. L'attributo “required” non è presente perché nella nostra convenzione abbiamo deciso che, se non diversamente specificato, sia impostato a true.

2° caso:

```
<attribute select="d_firmatario" name="ruolo">
        <check>isNotEmpty</check>
</attribute>
```

Il caso di un nodo di tipo Attr è leggermente diverso: con select viene sempre restituito l'insieme di nodi che soddisfa la query, mentre l'attributo name indica il nome dell'attributo su cui effettuare il controllo. Nell'esempio qui riportato, verrà controllato che l'attributo ruolo, del nodo d_firmatario, non abbia un valore nullo.

Qualora venga riscontrata una incongruenza nei campi editati

dall'utente, verrà inserito come figlio di quel nodo il contenuto del file XML che nel nel file di configurazione è associato a quel tipo di errore per quel determinato tag.
Ad esempio nel caso degli esempi vista prima, avremo, all'interno del file di configurazione, le righe:
xml.error.d_nome_ente.isNotEmpty=/errors/empty_field.xml
xml.error.d_firmatario.isNotEmpty=/errors/empty_field.xml

e il contenuto del file XML sarà un qualcosa del tipo:
<html>
il campo non può essere vuoto
</html>
A questo punto ci sono alcune osservazioni da fare a proposito dei vantaggi offerti dalla flessibilità del nostro framework nella fase di validazione:

- sebbene la classe Java che contiene i metodi di validazione sia già dotata di quelli più comuni, può essere necessario scrivere un ulteriore metodo di controllo per una determinata collection di documenti. In quel caso, sarà sufficiente estendere la classe comune con una che contiene il nuovo metodo.
- Qualora sia necessario effettuare dei controlli diversi su due nodi che hanno lo stesso nome, ma semantiche diverse (ad esempio il tag data può essere presente all'interno di protocollo che di firma) sarà sufficiente scrivere due triple nel file di configurazione con campi select che differiscono nel nodo padre e nel metodo da applicare e aggiungere i rispettivi mapping con i messaggi di errore.

Stampa in formato PDF

Una volta compilata (e verificata) la Comunicazione di Avvio relativa ad una domanda unica, oltre che a visualizzarla in formato XHTML, sarà anche possibile visualizzarla in formato PDF ed eventualmente

stamparla. Per usufruire di questa funzionalità del nostro framework ci è stato sufficiente scrivere un foglio di stile che trasformasse il documento XML della Comunicazione di Avvio in un XML conforme a XSL:FO, il quale, passato al modulo di visualizzazione in PDF insieme all'XML originale, lo trasforma in un documento PDF. C'è da osservare che qualora venga cambiato l'XML-Schema delle Comunicazioni di Avvio, sarà sufficiente riportare nel foglio di stile che lo trasforma in XSL:FO le eventuali modifiche perché la visualizzazione in PDF continui a funzionare.

Capitolo 7: Conclusioni e problemi rimasti aperti

Questo capitolo è dedicato alle conclusioni finali della tesi e ad alcuni problemi che sono rimasti senza soluzione.

Il framework che abbiamo progettato e sviluppato a supporto delle applicazioni per la gestione dello Sportello Unico per le Attività produttive si è rivelato abbastanza flessibile e generale da poter essere utilizzato per lo sviluppo di un qualsiasi servizio interattivo basato sul web che debba accedere ad una base documentale eterogenea e distribuita. Questo deriva dal fatto che, rispetto ad altri servizi interattivi, il SUAP gestisce tipi di documenti anche molto diversi tra loro, obbligando a realizzare delle soluzioni che siano il più generali possibile. Una funzionalità importante che purtroppo non è stato possibile implementare per mancanza di tempo è stata quella relativa alla firma digitale: anche se i PDF creati possono essere firmati attraverso l'uso di applicazioni di terze parti, questa soluzione non è integrata all'interno del framework e porta naturalmente a problemi di usabilità. Per quello che riguarda i problemi riscontrati durante lo sviluppo e che non hanno trovato soluzione, il principale è stato l'immaturità delle implementazioni di database XMLDB open source testate: Xindice ha un grave problema di gestione del formato dei caratteri, dato che taglia tutti i caratteri che hanno un codice ASCII superiore a 127 (comprese quindi le lettere accentate), mentre eXist non supporta ancora completamente lo standard XMLDB, specie per quanto riguarda le funzioni standard di XPath. Inoltre nessuno di questi due database supporta completamente lo standard Xquery, che consentirebbe un aumento di efficienza dovuto al minore numero di query necessarie al database. Infine, un problema minore, ma per cui comunque non esiste una soluzione generale, è quello dato dal fatto che la stessa pagina HTML si può presentare in maniera differente su browser diversi, dato che

non esiste un sottoinsieme standard di funzionalità comuni ai browser più diffusi che visualizzi i dati allo stesso modo e consenta di presentare le funzionalità necessarie ad un servizio interattivo.

Appendice A: note implementative

Class Diagram relativo a DBConnection

Questo diagramma illustra le relazioni tra le classi relative alla lettura e scrittura di documenti.

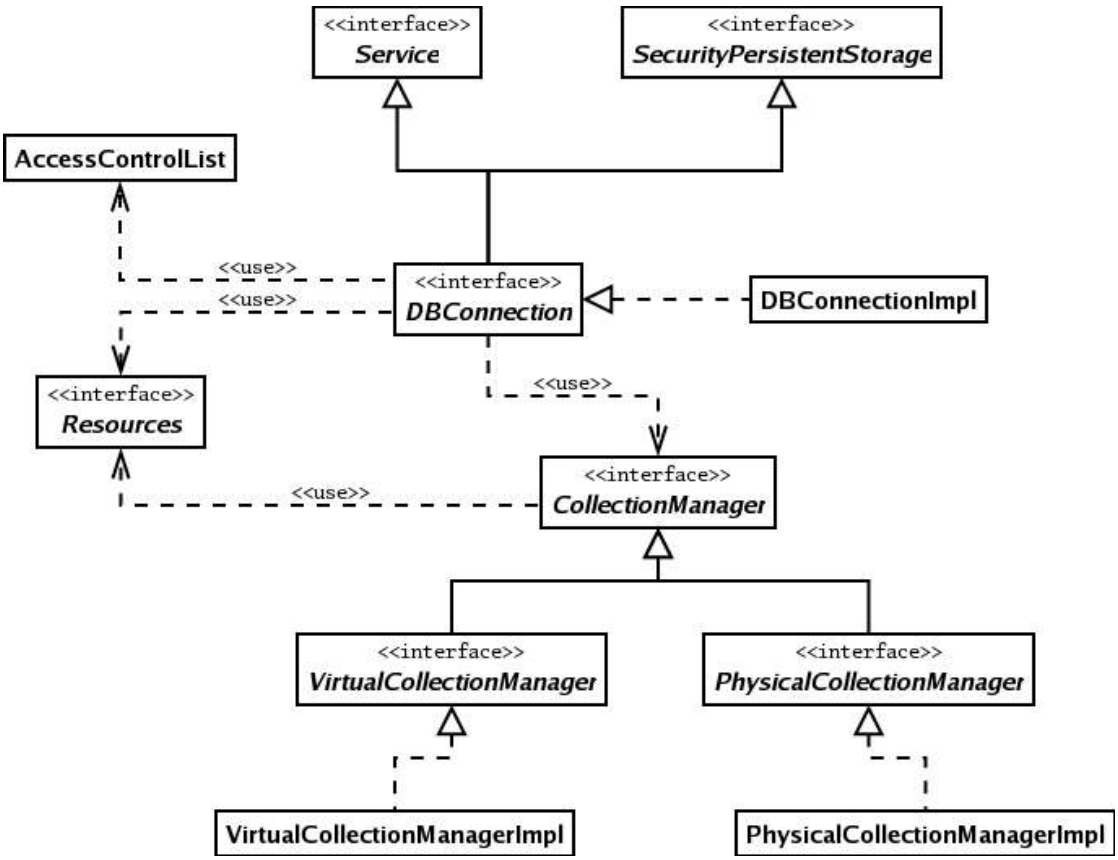

Figura 18: Diagramma delle classi di DBConnection

La Classe principale, che fornisce l'interfaccia alle applicazioni, è DBConnection. L'implementazione dell'interfaccia SecurityPersistentStorage indica che da DBConnection è anche possibile gestire le informazioni relative agli utenti. Sarebbe stato possibile trattare la gestione degli utenti in maniera non omogenea rispetto al resto dei documenti, ma questo avrebbe significato l'impossibilità di poter utilizzare il framework per sviluppare le applicazioni di gestione degli utenti. Per operare fisicamente sui

documenti, DBConnection non interagisce direttamente con il sistema, ma demanda il compito ad un oggetto che implementa l'interfaccia CollectionManager. Questa scelta è stata fatta perché oggetti di tipo diverso sono gestiti in modo differente: ad esempio un tipo di documenti potrebbe risiedere sul filesystem ed un altro potrebbe essere memorizzato in un database. La DBConnection, quindi, a seconda del tipo di documento gestito, utilizza un'implementazione diversa di CollectionManager per scrivere e leggere i documenti. CollectionManager si specializza in due sottointerfacce differenti che corrispondono alle due famiglie di tipi gestite dal framework: i tipi “fisici”, gestiti da PhysicalCollectionManager, sono quelli dei normali documenti, che esistono in un qualche repository indipendentemente dall'applicazione che li gestisce. I tipi “virtuali”, invece, non corrispondono a documenti esistenti, ma individuano documenti creati dinamicamente a partire da frammenti di documenti “fisici”. Oltre a gestire i CollectionManager, DBConnection si occupa di verificare i diritti dell'operazione richiesta attraverso la Access Control List dell'applicazione.

Activity Diagram di getObject (DBConnection)

Questo diagramma illustra la funzionalità di recupero documenti dell'interfaccia con la base di conoscenza.

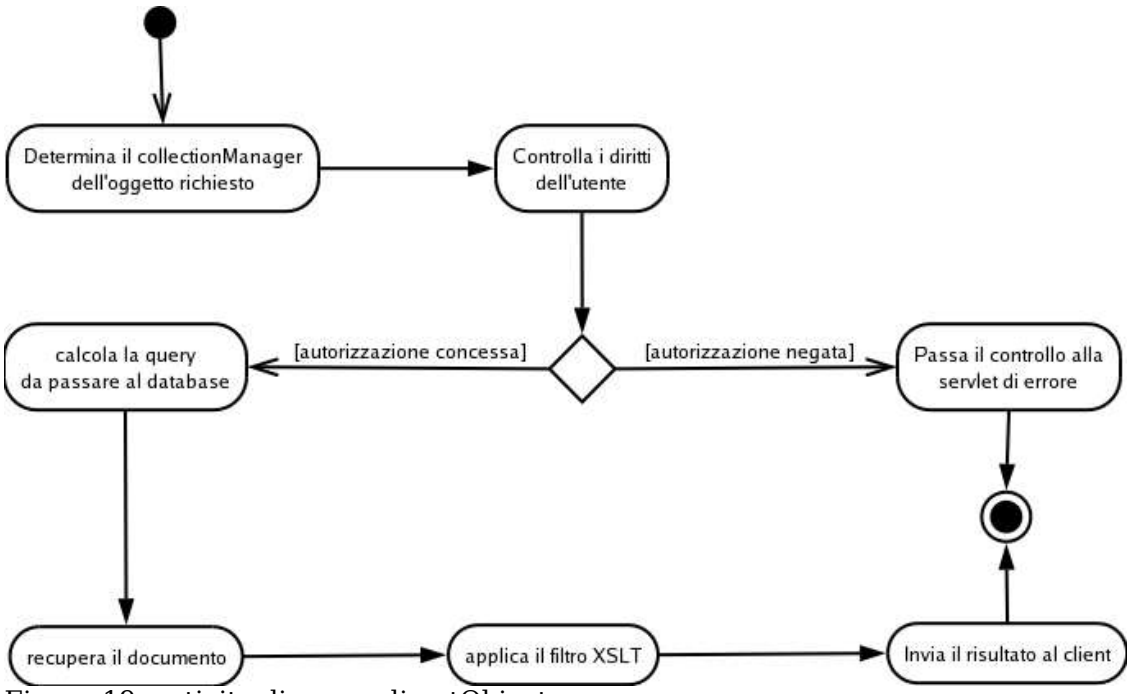

Figura 19: activity diagram di getObject

Il CollectionManager è un oggetto che varia a seconda del tipo di documento da recuperare e fornisce l'interfaccia attraverso la quale è possibile leggere o scrivere fisicamente i documenti di quel tipo. Infatti tipi diversi di documento possono avere modalità di accesso differenti: un esempio caratteristico è quello dei documenti “fisici” e di quelli “virtuali”. Nel primo caso il documento esiste fisicamente in un determinato repository e può essere recuperato in modo semplice; nel secondo caso il documento cercato è costruito a partire da diversi documenti (magari memorizzati in luoghi anche molto differenti, come un database e un filesystem) e la procedura di lettura del documento può diventare anche relativamente complessa. Utilizzando un'unica interfaccia che nasconde i dettagli di memorizzazione dei documenti, è possibile ignorare queste diversità

ed accedere in modo uniforme ad ogni tipo di documento.

Activity Diagram di NewObject (DBConnection)

Questo diagramma illustra la funzionalità di creazione di nuovi documenti dell'interfaccia con la base di conoscenza.

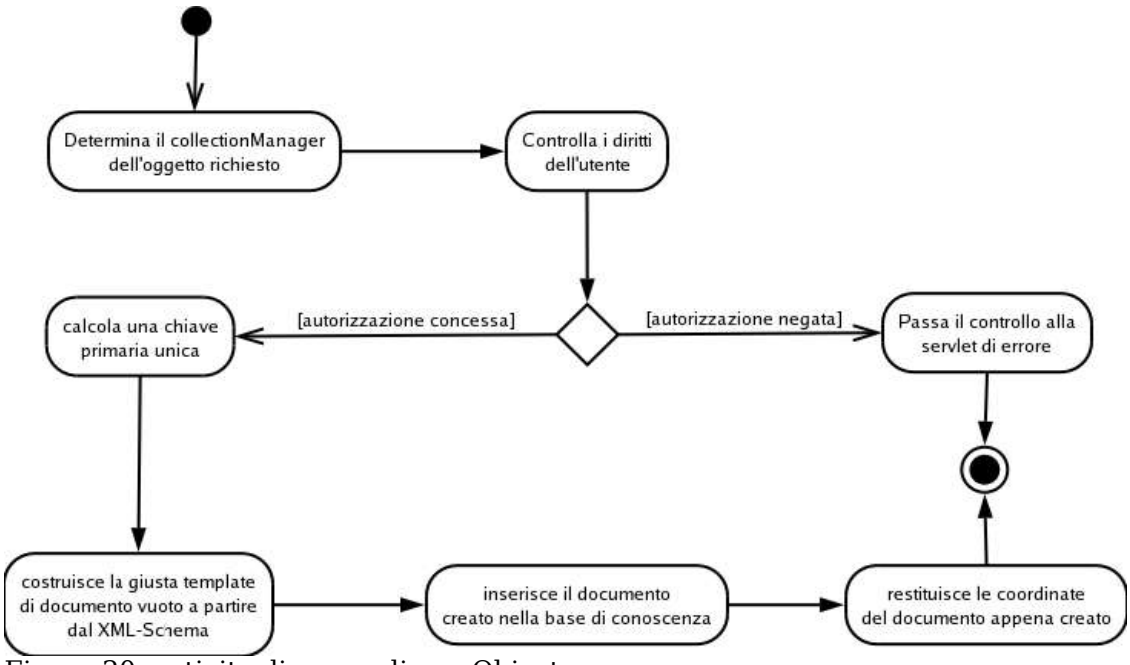

Figura 20: activity diagram di newObject

Il documento vuoto creato viene costruito attraverso l'applicazione di un filtro XSLT all'XML-Schema del documento stesso. Per questo documento minimale vengono costruiti solo i nodi indicati come non opzionali dallo Schema,in modo da ottenere il più piccolo documento possibile che abbia una struttura valida. Le coordinate restituite sono il tipo di documento e la chiave primaria.

Editing di un documento

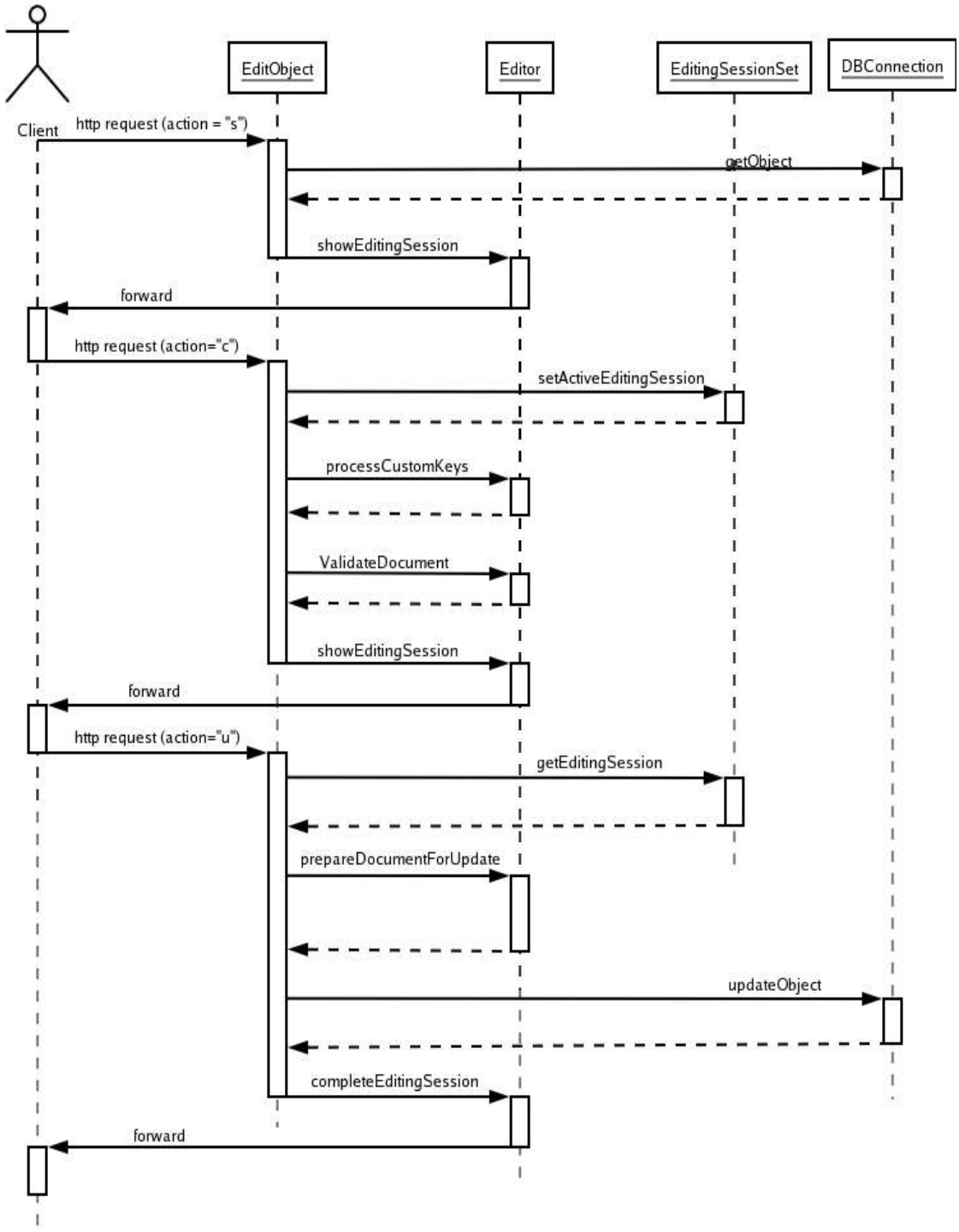

Figura 21: sequence diagram di editObject

Il diagramma descrive la sequenza delle operazioni che avvengono quando si edita un documento esistente nel caso più semplice in cui

non si apportano modifiche strutturali al documento e non ci sono errori di validazione. Il client web invia alla servlet di editing (attraverso la URL) le azioni da eseguire ed i loro parametri sono passati alla servlet di editing attraverso la URL inviata dal client web. Durante tutta l'operazione di modifica del documento e fino a che le modifiche non vengono confermate, il documento è mantenuto all'interno di un oggetto di sessione a causa del fatto che il protocollo HTTP non consente di mantenere uno stato durante la sessione. Le operazioni che agiscono su questa copia di lavoro del documento sono svolte da una classe che implementa l'interfaccia Editor. Oltre alla validazione e alla preparazione del documento per l'editing e per la memorizzazione definitiva all'interno della base di conoscenza, l'altra funzionalità importante data da Editor è la restituzione del controllo al client attraverso ShowEditingSession. Questo metodo fornisce una funzionalità simile a quella presente nella servlet di visualizzazione (ViewObject), e trasforma, attraverso un foglio di stile XSLT, il documento XML in una pagina XHTML contenente le maschere di inserimento già compilate con i dati contenuti nel documento. La funzionalità ProcessCustomKeys, nonostante non sia una delle funzioni principali di Editor, è stata inserita nel diagramma perché mostra che nell'editing dei documenti esistono casi particolari che non possono essere gestiti in maniera generale dal framework: quest'ultimo, infatti, basa la metodologia di editing sul fatto che vi sia una corrispondenza univoca tra una maschera di inserimento e un nodo del documento XML. Purtroppo esistono casi in cui questa assunzione non può essere soddisfatta, come ad esempio il caso in cui si debbano modificare o creare delle password. In questo caso, infatti, la prassi comunemente seguita è quella di avere due maschere di inserimento identiche in ognuna delle quali l'utente scrive “alla cieca” la sua password, in modo da avere una ragionevole sicurezza di aver inserito la password giusta. Questi due campi, però, corrispondono al medesimo nodo del documento che

rappresenta l'utente, e quindi non possono essere gestiti in maniera omogenea agli altri. ProcessCustomKeys si occupa quindi di gestire questi casi particolari. Un'ultima cosa da notare è che durante una sessione di editing è possibile che vengano editati più documenti, e quindi un unico oggetto di sessione è insufficiente. Per questo nel diagramma è presente un insieme di oggetti di sessione (EditingSessionSet). Come si può notare dal diagramma di sequenza, l'operazione di editing è divisa (di base) in tre fasi:

1) recupero documento (azione “s”)
2) editing documento (azione “c”)
3) memorizzazione documento (azione “u”)

Il punto (2) può ripetersi un numero indeterminato di volte, fino a che l'utente non decide che le modifiche apportate sono soddisfacenti, e può presentare alcune variazioni rispetto al caso base illustrato sopra.

Errori di validazione

In questo diagramma di sequenza viene illustrato il caso in cui sia presente un errore di validazione rispetto ai dati inseriti dall'utente.

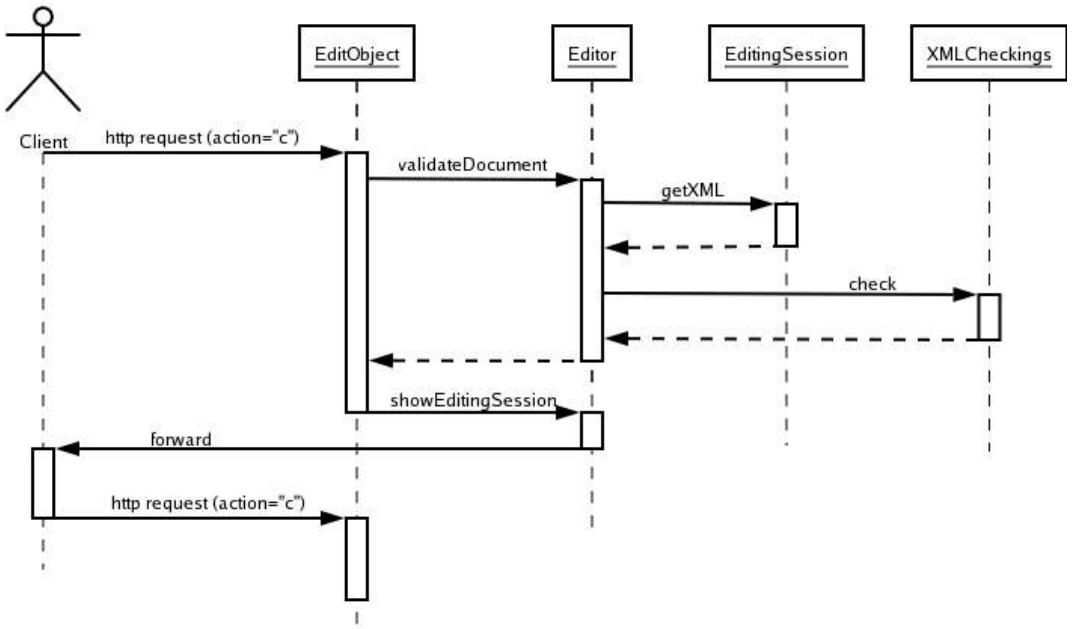

Figura 22: errore di validazione di getObject

Al contrario del caso generale, si sono resi visibili nel diagramma sia l'oggetto di sessione che contiene il documento, che la classe che contiene i metodi di verifica del documento. Ad ogni nodo il cui contenuto non supera la verifica viene aggiunto un nodo figlio contenente un messaggio di errore Questi nodi vengono gestiti dal foglio di stile XSLT utilizzato da ShowEditingSession, il quale si occupa di visualizzare gli eventuali errori vicino alle corrispondenti maschere di inserimento dati.

Modifica della struttura del XML:

In questo caso viene illustrato il caso in cui la struttura dell'XML venga modificata. Per alcuni tipi di documento, infatti, è possibile

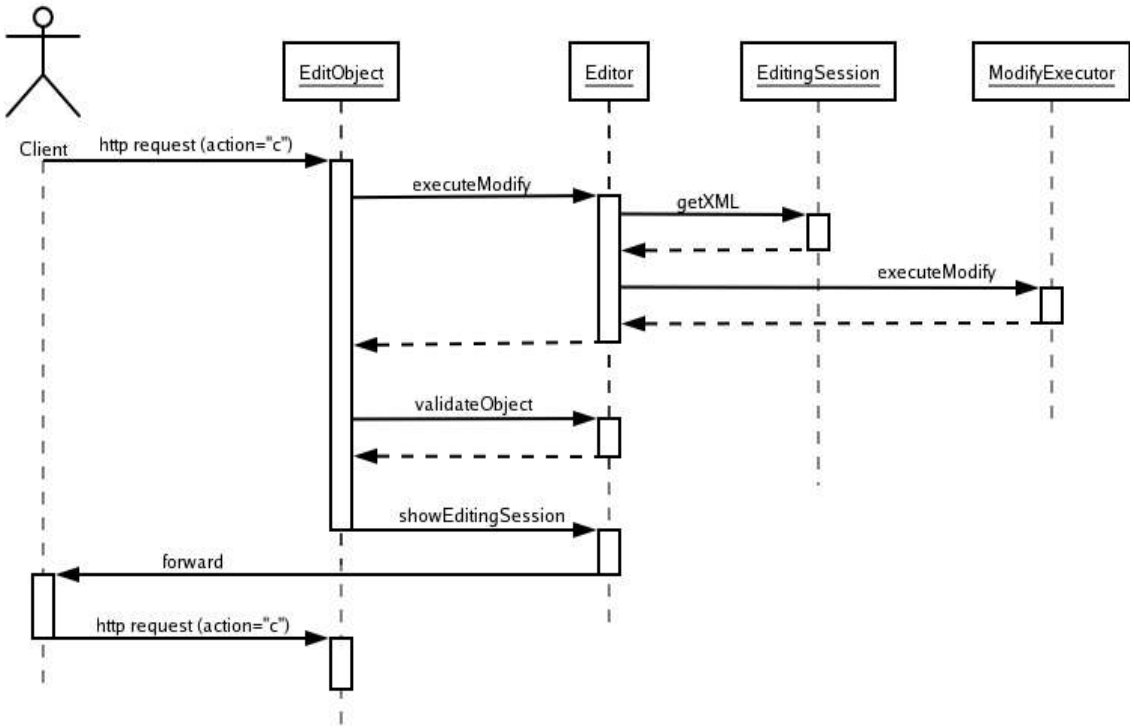

Figura 23: modifica della struttura del documento

che alcuni campi siano opzionali o di cardinalità superiore a uno. Deve quindi essere possibile modificare la struttura del documento in modo che si possano aggiungere o togliere dei campi. Per questo è stata progettata l'interfaccia ModifyExecutor, le cui implementazioni si occupano proprio di aggiungere e togliere nodi al documento senza invalidarlo rispetto al XML-Schema.

Redirezione

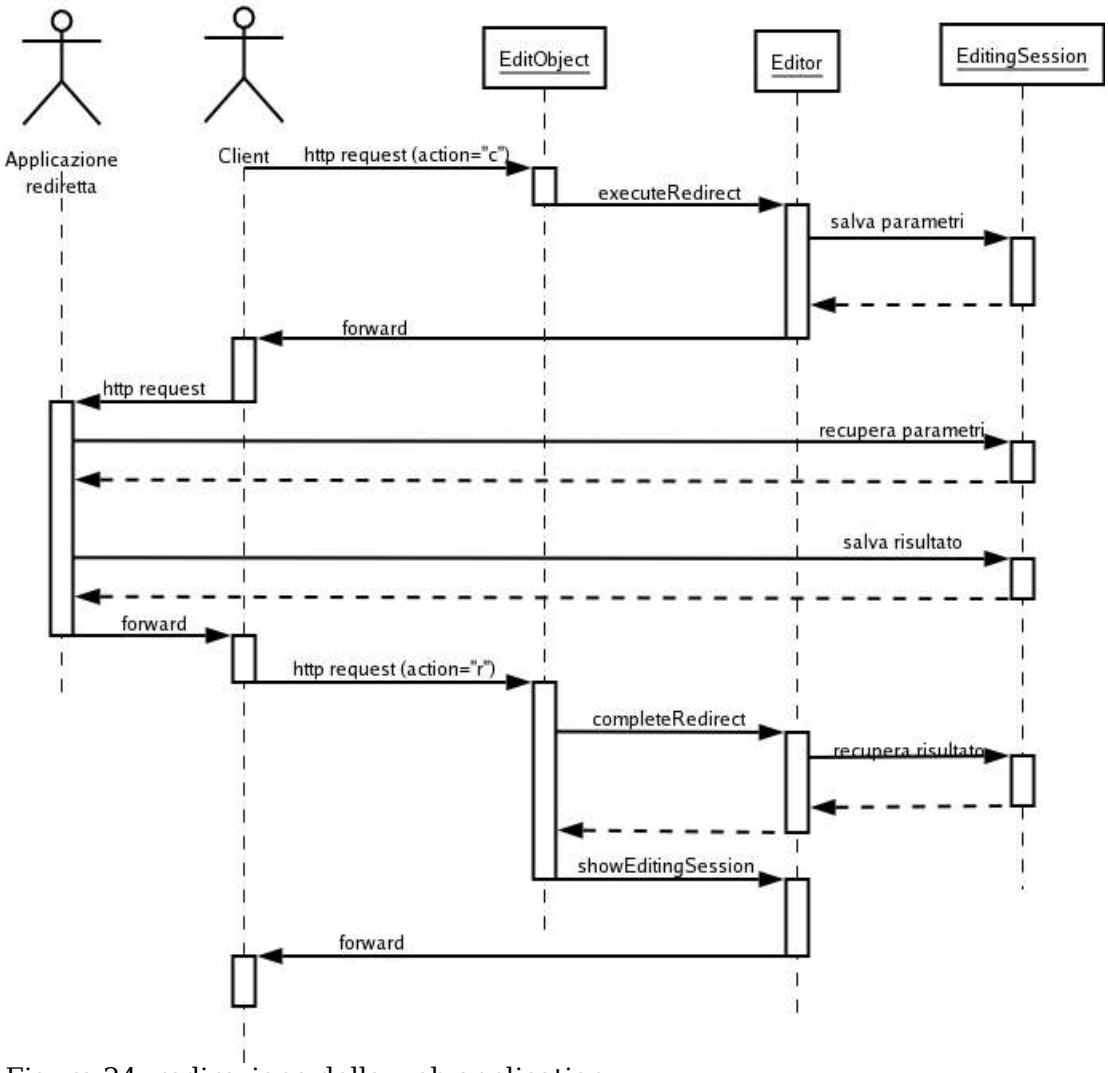

Figura 24: redirezione della web application

In questo diagramma viene illustrato il caso in cui il flusso di editing viene rediretto ad un'altra applicazione. I parametri ed i valori di ritorno vengono passati da un'applicazione web all'altra attraverso l'oggetto di sessione.

Collaboration Diagram di getObject (DBConnection)

Questo diagramma illustra il funzionamento della servlet di visualizzazione documenti. Il diagramma non ha bisogno di note particolari.

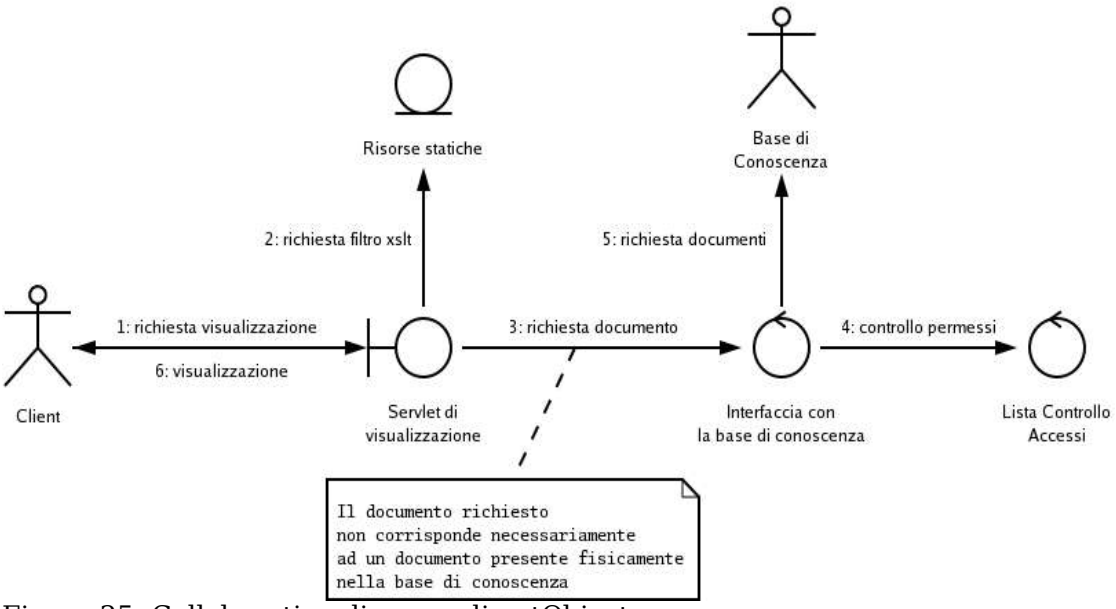

Figura 25: Collaboration diagram di getObject

Class Diagram relativo a SecurityManager

Questo diagramma illustra le relazioni tra le classi responsabili della gestione degli utenti.

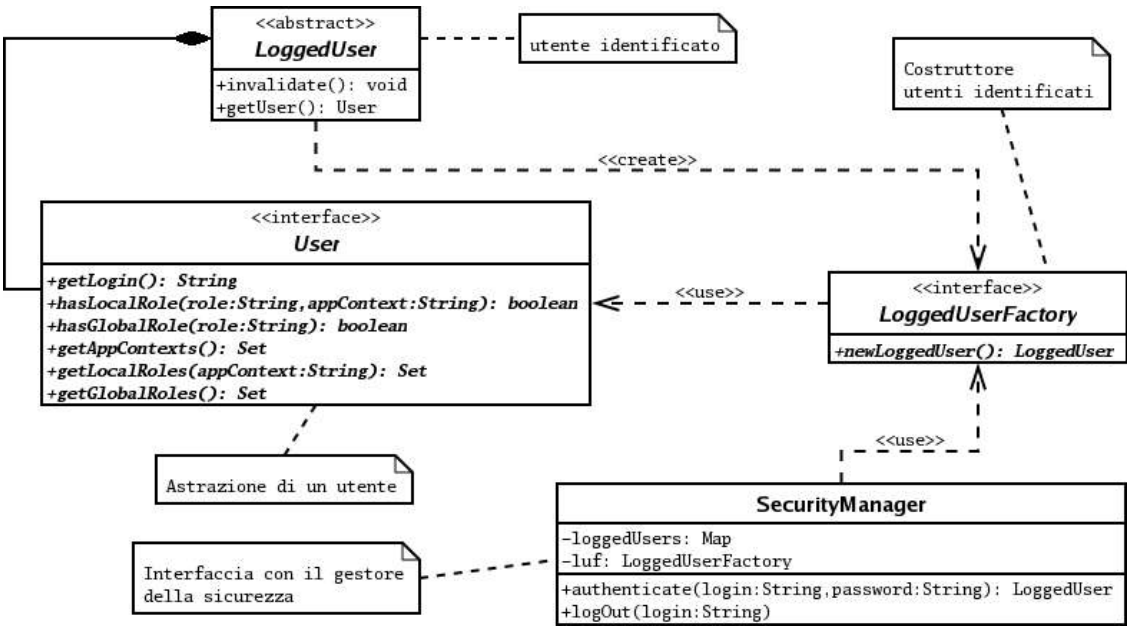

Figura 26: class diagram di SecurityManager

La classe che si interfaccia con l'applicazione è SecurityManager. Questa classe fornisce un oggetto di tipo LoggedUser (creato attraverso il pattern Factory da LoggedUserfactory) che individua l'utente che sta accedendo alla base di conoscenza. LoggedUser contiene un'istanza di User con le informazioni sull'utente.

Coll. diagram relativo alla procedura di autenticazione

Questo diagramma illustra la fase di autenticazione dell'utente.

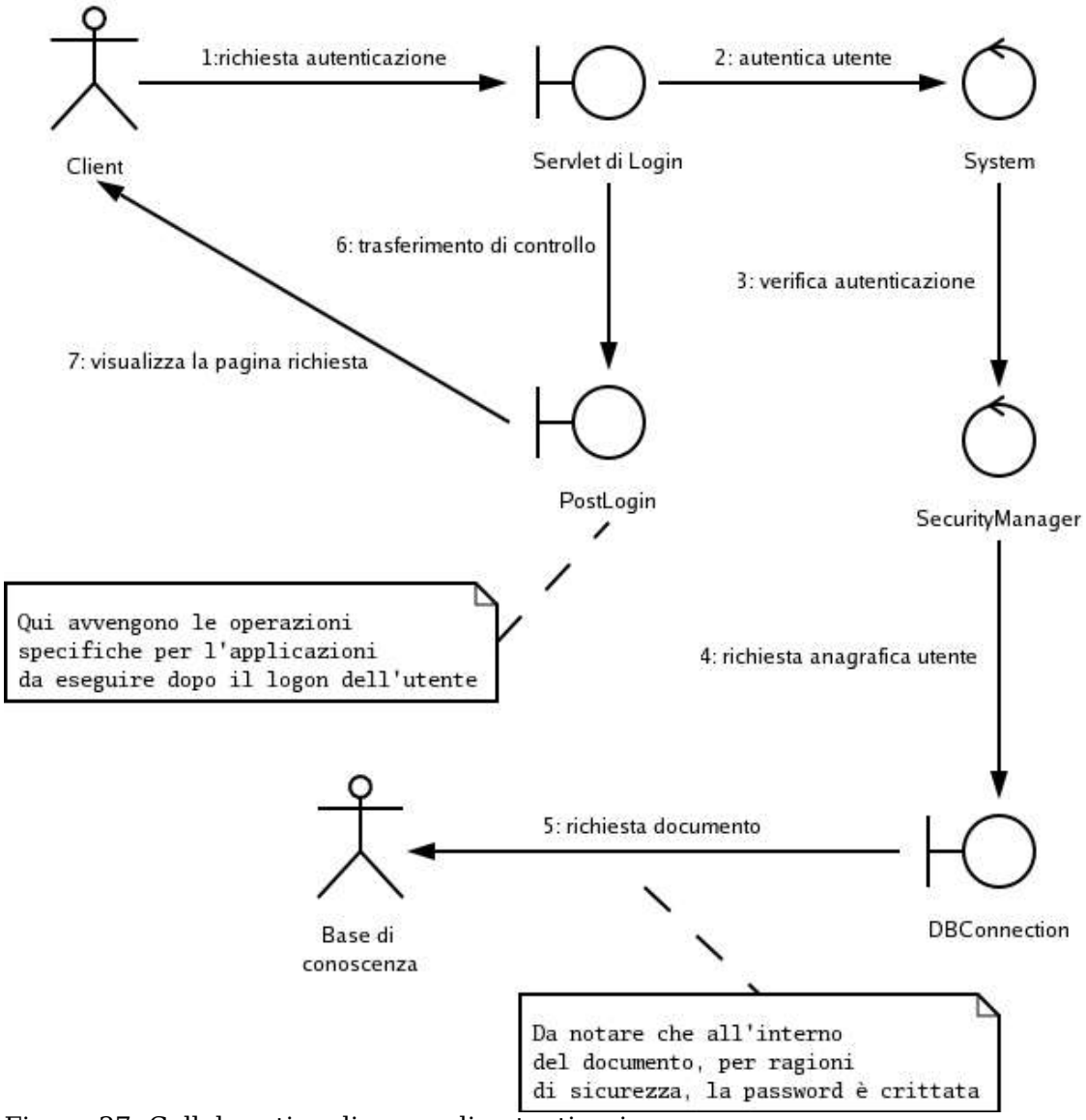

Figura 27: Collaboration diagram di autenticazione

Le informazioni relative all'utente sono inserite in un documento presente all'interno della base di conoscenza, che viene recuperato dalla DBConnection in maniera analoga a tutti gli altri documenti. La servlet PostLogin effettua tutte le operazioni occorrenti una volta che l'utente è stato identificato. In particolare si occupa di calcolare la

pagina verso cui viene rediretto il client. Questa pagina non può essere stabilita a priori, dato che l'applicazione può aver bisogno di autenticare l'utente in un qualsiasi punto (ad esempio nel caso che sia scaduta la sessione http) ed è necessario che ad autenticazione avvenuta l'applicazione web visualizzi la pagina relativa a quel punto, e non una generica pagina di benvenuto.

Collaboration Diagram relativo a Resources

La classe Resources fornisce l'interfaccia con tutte le risorse statiche del sistema.

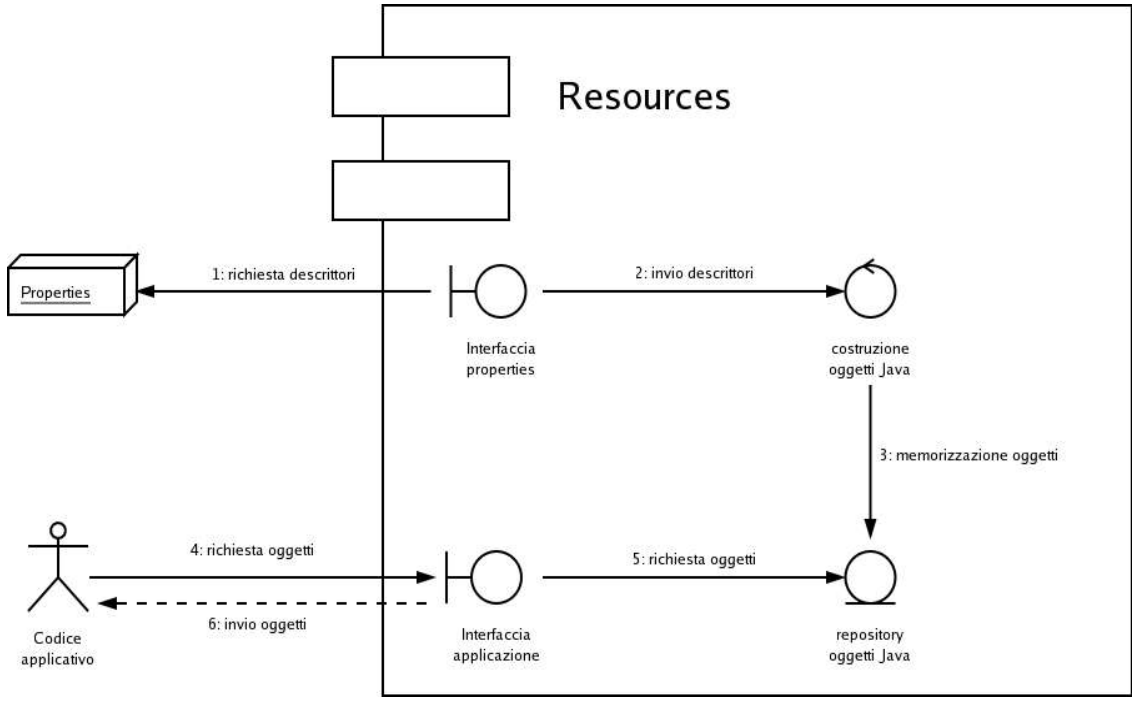

Figura 28: Collaboration diagram di Resources

I descrittori delle risorse da creare (principalmente nomi di file o di classi) non sono inseriti all'interno del codice della classe, ma sono memorizzati all'interno dei file di properties (il metodo standard di Java per la gestione delle configurazioni). Resources richiede questi descrittori e crea direttamente degli oggetti Java che astraggono queste risorse. Questi oggetti vengono creati una sola volta, durante l'inizializzazione di Resources, e sono memorizzati in attesa di essere richiesti dal codice applicativo. Questo porta certamente ad un utilizzo maggiore della memoria ma, vista la relativamente scarsa dimensione delle risorse e considerati sia il tempo medio di accesso al filesystem (per le risorse che vengono caricate da file), sia il costo della creazione di un nuovo oggetto, questo è stato ritenuto un compromesso accettabile. La scelta di creare direttamente degli

oggetti permette di utilizzare immediatamente le risorse, senza dover operare ulteriori trasformazioni: ad esempio, per i fogli di stile XSLT, quelli che vengono memorizzati e restituiti non sono gli XML che li rappresentano, ma degli oggetti di trasformazione (“Transformer”) che si incaricano della trasformazione vera e propria per quel determinato foglio di stile. In questo caso il guadagno di efficienza dato dal non dover creare un oggetto di trasformazione ogni volta che si deve visualizzare un documento viene ad essere notevole, vista la frequenza con cui questa operazione viene solitamente compiuta.

Gestione degli errori

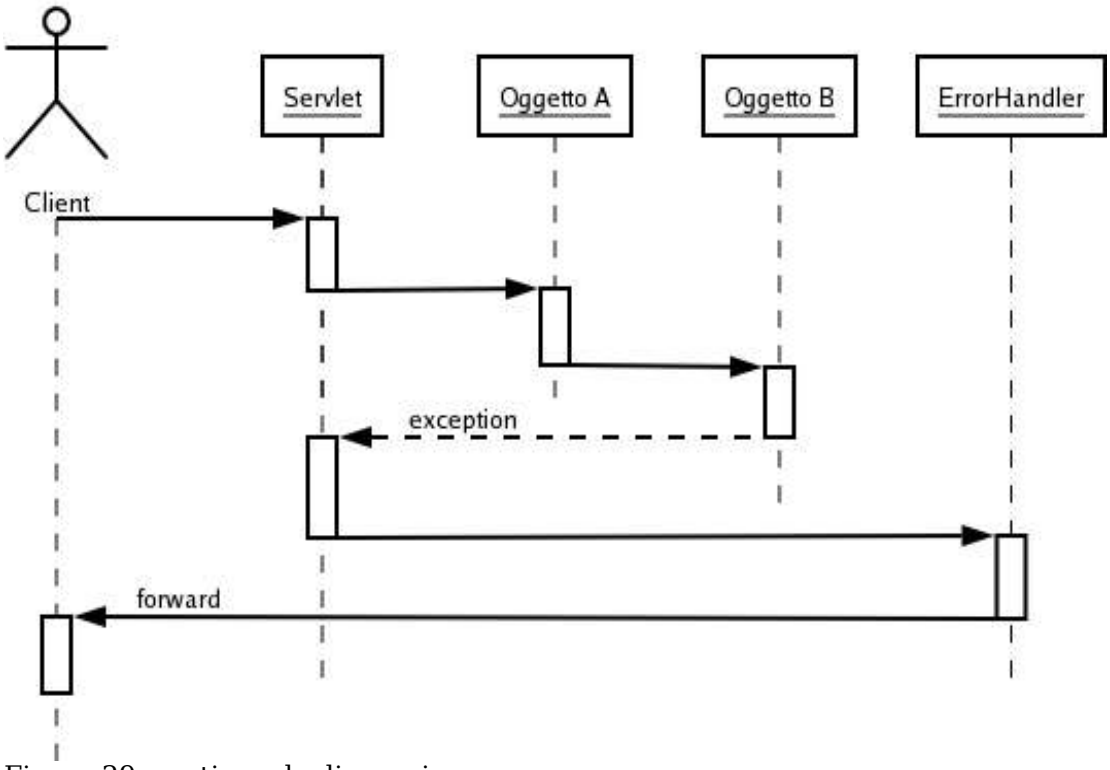

Figura 29: gestione degli errori

Il meccanismo di gestione degli errori realizzato nel framework è allo stesso tempo semplice e flessibile. Sfruttando la capacità di Java di demandare la gestione di una situazione di errore al metodo chiamante, la gestione degli errori non recuperabili viene centralizzata all'interno delle servlet, dove un singolo costrutto try/catch viene utilizzato per intercettare ogni genere di errore e per passare il controllo al gestore delle eccezioni adatto (una classe che implementa l'interfaccia ErrorHandler). Ogni servlet ha un gestore che viene utilizzato di default, ma è possibile specificare un gestore differente attraverso uno specifico parametro nell'URL di invocazione della servlet. L'implementazione più semplice del gestore si limita a inserire l'identificativo dell'errore all'interno di un Java Bean, lasciando la presentazione dell'errore al client che, attraverso il valore contenuto nel Bean, visualizza il messaggio di

errore adatto all'utente. Infine, questo meccanismo viene utilizzato senza alcuna modifica anche nel caso in cui si usino dei Custom Tag all'interno di pagine JSP. In questo caso il costrutto try/catch è inserito all'interno del Custom Tag.

Interfaccia verso l'utente

L'idea principale che ha guidato lo sviluppo dell'interfaccia è stata quella di avere una pagina web che visualizzasse assieme parti fisse (contenenti, ad esempio, le varie intestazioni) e parti variabili, che presentavano un contenuto diverso a seconda delle operazioni effettuate dall'utente. Una maniera semplice per ottenere questo risultato poteva essere quello di utilizzare dei frame per ogni sottoblocco della pagina, gestendo ognuno di questi in modo statico o dinamico, ma abbiamo preferito scartare questa soluzione. La ragione principale di questa scelta consiste nel fatto che per gestire la sincronizzazione tra frame differenti avremmo dovuto complicare notevolmente la web application. La soluzione adottata, invece, consiste nell'utilizzare un'unica pagina JSP che viene renderizzata in XHTML in modo diverso a seconda del contenuto di un bean di sessione gestito dai moduli della web application (vedi figura 30).

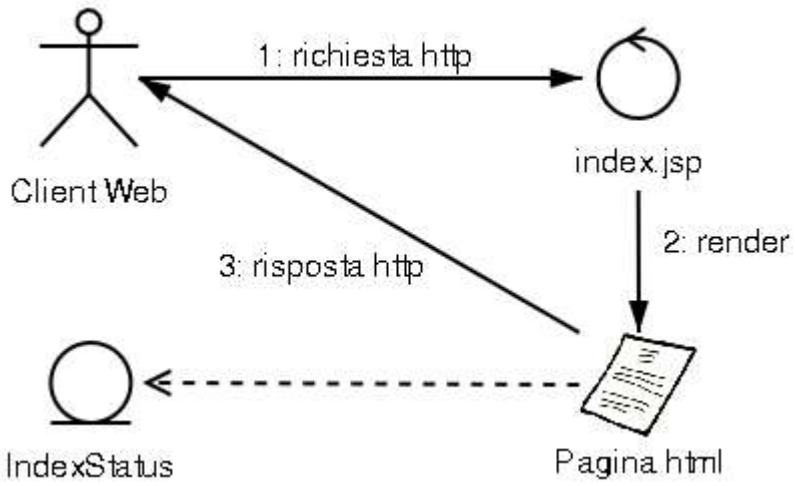

Figura 30: collaboration diagram index.jsp

Le parti fisse del portale sono state inserite direttamente nella JSP come codice XHTML, mentre per quelle che dipendono dal contesto si è utilizzato il costrutto if messo a disposizione dalle JSP per scegliere un contenuto da visualizzare. Per ragioni di pulizia del

codice, la pagina principale è solo uno scheletro che include le diverse sottopagine che costituiscono il portale attraverso la direttiva include delle JSP. La struttura di index.jsp è descritta in figura 31.

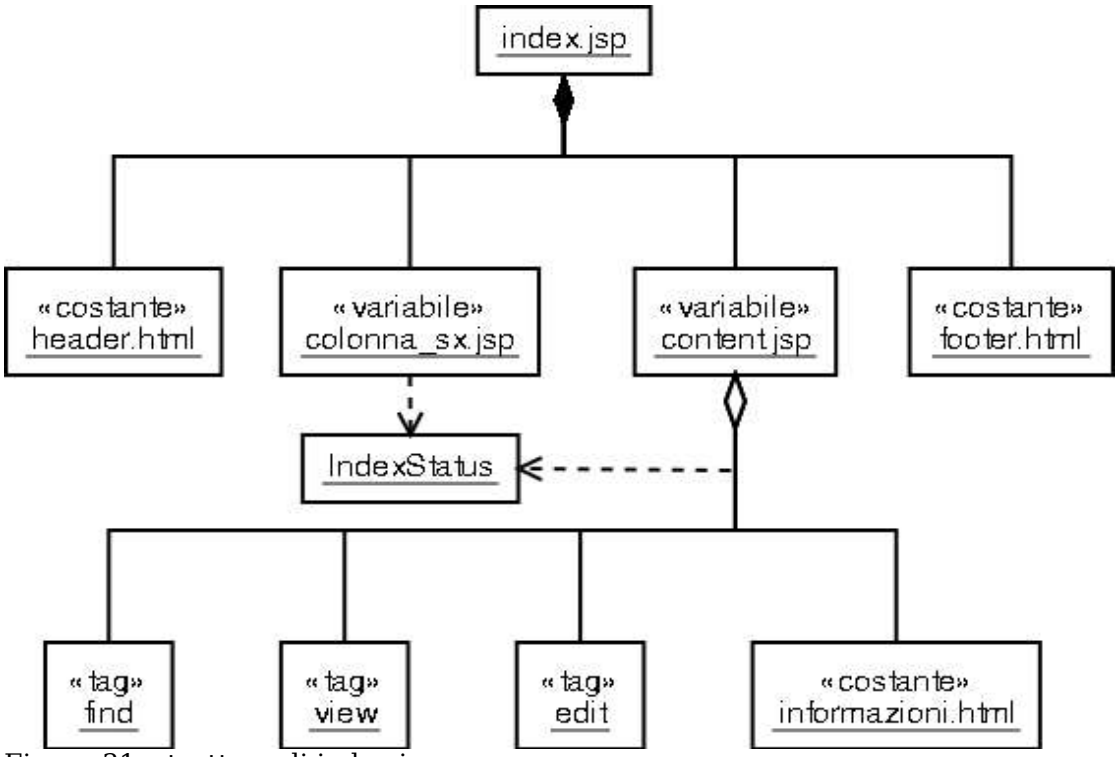

Figura 31: struttura di index.jsp

Le template “costante” e “variabile” si riferiscono al fatto che la pagina contenga o meno del codice JSP che visualizza un contenuto diverso a seconda del contesto. La template “tag” indica un custom tag che racchiude una funzionalità del framework. Header.jsp e footer.jsp sono semplicemente l'intestazione e il pié di pagina inseriti in ogni pagina html. Colonna_sx.jsp è la colonna con i menù e l'eventuale form di login. Quest'ultima è presente solo se l'utente che visita la pagina non si è ancora autentificato (questa informazione è presente in IndexStatus). Content.jsp, infine, contiene un unico costrutto condizionale che visualizza un custom tag o un frammento html a seconda del valore di un determinato campo di IndexStatus, il quale è utilizzato dalla web application per indicare lo stato attuale

di visualizzazione.

Appendice B: statistiche sul codice

In tabella sono riportate alcune statistiche relative al codice sviluppato:

modulo	n. classi	Linee di codice
utility	78	8745
Security	29	1148
Usereditor	24	3309
xmleditor		
db	27	3362
Editor	15	1482
error_handler	7	378
Tagext	7	982
Servlet	7	1519
Session	3	192
Resources	1	296
System	1	437
Totale xmleditor	68	8648
Totale framework	199	21850

Tabella 5: informazioni sul codice

Bibliografia

[Savona03] Savona et al. : Rapporto innovazione e tecnologie digitali in Italia - 2003

[ConsMin02] Presidenza del consiglio dei ministri : Front office e servizi di e-government per cittadini e imprese - 2002

[DPP447] Gazzetta Ufficiale della Repubblica Italiana - decreto del presidente della Repubblica 20 ottobre 1998 n. 447

[FerraioloKuhn92] David Ferraiolo, Richard Kuhn : 15th National Computer Security Conference - Role-Based Access Controls

[Schneider03] Fred B. Schneider : IEEE Security and Privacy - Least Privilege and more

[Ambriola02] Vincenzo Ambriola, Silvia Bertagnini, Letizia Pratesi : Pubblica amministrazione on line - 2002

[Bevilacqua02] Nicoletta Bevilacqua : E-Goverment: nuovi aradigmi organizzativi e formativi nelle regioni e negli enti locali - 2002

[JoySteeleGoslingBracha00] Bill Joy, Guy Steele, JamesGosling, Gilad Bracha - The Java Language Specification, Second Edition

[ArnoldGoslingHolmes00] Ken Arnold, James Gosling, David Holmes - The Java Programming Language, Third Edition

[Obasanjo01] Dare Obasanjo - A Survey of APIs and Techniques for Processing XML

[Harold2001] Elliotte Harold - XML Bible

[Bourret03] Ronald Bourret - XML and Databases

[Trinidad00] Gerardo Trinidad : Proceedings of the Philippines Computing Science Congress 2000 - XML and Databases for Internet Applications

[Kellokoski00] Kari Kellokoski : Tik-111.590 Research Seminar on Digital Media fall 2000: XML - XML Repositories

[XMLDBFAQ] XML:DB Frequently Asked Questions (http://www.xmldb.org/faqs.html)

[Staken03] Kimbro Staken - Introduction to Native XML Databases

[XindiceTech] Xindice Internals Guide (http://xml.apache.org/xindice/dev/guide-internals.html)

[Meier02] Wolfgang Meier : Web, Web-Services, and Database Systems. NODe 2002 Web- and Database-Related Workshops - eXist: An Open Source Native XML Database

[Li01] Sing Li - The Tomcat story

[Apache02] : Apache Software Foundation,
(http://jakarta.apache.org/tomcat/tomcat-4.1-doc/ssl-howto.html)

[Coward01] Java Servlet 2.3 Specification

[Armstrong03] Eric Armstrong et al. - The Java Web Services Tutorial

[Graham97] Hamilton Graham - JavaBeans Specifications

[Pelegri01] Eduardo Pelegri - JavaServer Pages 1.2 Specification

[Mamei00] Marco Mamei - Sviluppo di applicazioni Internet: l'uso integrato di XML e Java

[Gushee03] Matt Gushee et al. - XSL History

[Deach02] Stephen Deach - What Is XSL-FO and When Should I Use It?

[Froumentin00] Max Froumentin - History of XSL

[Cover03] Robin Cover - Extensible Stylesheet Language (XSL)

[ClarkDeRose99] James Clark, Steve DeRose - XML Path Language (XPath) Version 1.0

[Unidex01] Unidex - Universal Turing Machine in XSLT

[Novatchev02] Dimitre Novatchev - Dynamic Functions using FXSL: Composition, Partial Applications and Lambda Expressions

[Clark99] James Clark - XSL Transformations (XSLT) Version 1.0

[Adler01] Sharon Adler et al. - W3C Recommendation 15 October 2001

[CNIPA02] Carlo Batini : Formato per la rappresentazione elettronica dei provvedimenti normativi tramite illinguaggio di marcatura XML - 2002

[Kartchner98] Chris Kartchner : The Journal of Electronic Publishing - Getting from Concept to Reality

[Sydanmaanlakka00] Pentti Sydanmaanlakka : 2nd Annual Knowledge Management and Organizational Learning Conference - Understanding Organizational Learning through Knowledge Management, Competence Management, and Performance Management